

DTIC FILE COPY

ESC 1954

2

AFATL-TR-89-20

# Computation of Transonic Flow About Stores

---

David L Whitfield

Joe F Thompson

MISSISSIPPI STATE UNIVERSITY  
DEPARTMENT OF AEROSPACE ENGINEERING  
MISSISSIPPI STATE, MS 39762

JUNE 1989

FINAL REPORT FOR PERIOD JUNE 1984 - SEPTEMBER 1988

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC  
ELECTE  
JUL 0 8 1989  
S E D

AIR FORCE ARMAMENT LABORATORY  
Air Force Systems Command ■ United States Air Force ■ Eglin Air Force Base, Florida

99 05 068

AD-A210 402

## NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The AFATL STINFO program manager has reviewed this report, and it is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

*Stephen C. Korn*

STEPHEN C. KORN  
Technical Director, Aeromechanics Division

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization, please notify AFATL/FXA, Eglin AFB FL 32542-5434.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE			4. MONITORING ORGANIZATION REPORT NUMBER(S) AFATL-TR-89-20		
4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A					
6a. NAME OF PERFORMING ORGANIZATION Mississippi State University		6b. OFFICE SYMBOL (If applicable) N/A		7a. NAME OF MONITORING ORGANIZATION Aerodynamics Branch Aeromechanics Division	
6c. ADDRESS (City, State, and ZIP Code) Department of Aerospace Engineering Drawer A Mississippi State, MS 39762		7b. ADDRESS (City, State, and ZIP Code) Air Force Armament Laboratory Eglin AFB FL 32542-5434			
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Aeromechanics Division		8b. OFFICE SYMBOL (If applicable) AFATL/FX		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F08635-84-C-0228	
8c. ADDRESS (City, State, and ZIP Code) Air Force Armament Laboratory Eglin AFB FL 32542-5434		10. SOURCE OF FUNDING NUMBERS			
		PROGRAM ELEMENT NO. 61102F		PROJECT NO. 2307	TASK NO. E1
				WORK UNIT ACCESSION NO. 24	
11. TITLE (Include Security Classification) Computation of Transonic Flow About Stores					
12. PERSONAL AUTHOR(S) David L. Whitfield and Joe F. Thompson					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM Jun 84 to Sep 88		14. DATE OF REPORT (Year, Month, Day) June 1989	
15. PAGE COUNT 55					
16. SUPPLEMENTARY NOTATION Availability of this report is specified on verso of front cover.					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Store Separation		
01	01		Numerical Grid Generation Euler Equations		
			Computational Fluid Dynamics Navier-Stokes Equations		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) The objective of this program was to derive grid generation and fluid flow algorithms to predict the aerodynamics of aircraft/weapon configurations. Both captive carriage and the launch transient mode were considered in the design of the methods. The numerical grid generation techniques developed were both algebraic, based on transfinite interpolation, and elliptic, based on iterative solution of partial differential equations. The techniques were incorporated into one code that became the numerical grid generation for the Eglin Arbitrary Geometry Implicit Euler (EAGLE) code. In addition, an adaptive version of the grid generation code was developed and applied to missile configurations. The flow solvers developed were all upwind, finite volume schemes ranging from explicit, split-flux vector to implicit, split-flux difference algorithms. Both steady and unsteady, Euler and Navier-Stokes, were written and applied to various configurations. Two implicit algorithms were employed to form the EAGLE flow solver code. This code was run on complex aircraft/weapon configurations, including the launch transient problem of a weapon releasing from an aircraft pylon.					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Lawrence E. Lijewski			22b. TELEPHONE (Include Area Code) 904-882-3124		22c. OFFICE SYMBOL AFATL/FXA

DD Form 1473, JUN 86

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

# PREFACE

This program was conducted by the Mississippi State University, Starkville, Mississippi, under Contract F08635-84-C-0228, with the Air Force Armament Laboratory, Eglin Air Force Base, Florida 32542-5434. Dr. Lawrence E. Lijewski, AFATL/FXA, managed the program for the Armament Laboratory. The principal investigators were Dr. David L. Whitfield and Dr. Joe F. Thompson, Mississippi State University, Mississippi State, Mississippi 39762. The program was conducted during the period from June 1984 through September 1988.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



# TABLE OF CONTENTS

Section	Title	Page
I	INTRODUCTION.....	1
II	FLOW SOLVERS.....	2
	1. Equation Formulation.....	2
	2. Numerical Algorithms Investigated: A History and Perspective of SKOAL, COPEN, BMULE, and REDOX.....	3
	3. Results.....	7
	4. Conclusions.....	22
III	GRID GENERATION SYSTEM.....	22
	1. Generation System Formulation.....	22
	2. Code Foundation.....	28
	3. Results.....	33
	4. Conclusions.....	42
IV	RECOMMENDATIONS.....	44
	1. Flow Solvers.....	44
	2. Grid Generation System.....	44
	REFERENCES.....	46

# LIST OF FIGURES

Figure	Title	Page
1	Wing-Pylon-Store Configuration.....	10
2	Wing-Pylon-Store Configuration Showing Locations of Numerical and Experimental Comparisons.....	11
3	Numerical and Experimental Surface Pressure Distributions on the Outboard Side of the Store at $M=0.85$ , $\alpha=0.0$ .....	12
4	Numerical and Experimental Surface Pressure Distributions on the Inboard Side of the Store at $M=0.85$ , $\alpha=0.0$ .....	13
5	Numerical and Experimental Surface Pressure Distributions on the Pylon (Lower Row, $Y=0.67$ ) at $M=0.85$ , $\alpha=0.0$ (Captive Position).....	14
6	Numerical and Experimental Surface Pressure Distributions on the Pylon (Upper Row, $Y=1.17$ ) at $M=0.85$ , $\alpha=0.0$ (Captive Position).....	15
7	Numerical and Experimental Surface Pressure Distributions on the Pylon (Lower Row, $Y=0.67$ ) at $M=0.85$ , $\alpha=0.0$ (Released Position).....	16
8	Numerical and Experimental Surface Pressure Distributions on the Pylon (Upper Row, $Y=1.17$ ) at $M=0.85$ , $\alpha=0.0$ (Released Position).....	17
9	Computed Surface Pressure Contours with Store Located in Captive Position and Moving Through 0.4, 1.0, and 2.0 Store Diameters Below the Pylon - View 1.....	18
10	Computed Surface Pressure Contours with Store Located in Captive Position and Moving Through 0.4, 1.0, and 2.0 Store Diameters Below the Pylon - View 2.....	20
11	Computed Surface Pressure Contours with Store Located in Captive Position and Moving Through 0.4, 1.0, and 2.0 Store Diameters Below the Pylon - View 3.....	21
12	C-O Grid for Wing-Pylon-Store.....	36

## SECTION I

### INTRODUCTION

This is the final report of a 40-month research contract with the Armament Laboratory, Eglin Air Force Base, FL, aimed at the development of grid generation techniques and computational methods for predicting transonic flow about stores in the captive and vertical launch positions. This research was carried out through a very integrated and cooperative effort between personnel at Eglin AFB and Mississippi State University. The total research effort in store aerodynamics produced two MS theses (References 1 and 2), four PhD dissertations (References 3 through 6) that are completed, and three additional PhD dissertations (References 7 through 9) that are expected to be completed this school year. Moreover, this research produced or contributed to 36 other publications (References 10 through 45). It also led to the development of the Air Force EAGLE Code, which has been widely distributed to agencies in the Department of Defense (References 18,19,41).

The purpose of this report is not to cover the details of all the work performed, which would not be practical as indicated by the large number of publications given above. Rather, the purpose is to summarize the complete research effort and to try and tie together the total program. This research is more or less naturally divided into two parts, computational methods for solving the flow equations and grid generation techniques for constructing meshes. Consequently the report is divided into two technical areas.

The first area, Section II, of the report covers the computational methods for solving the flow equations. To this end, Paragraph 1 covers the equations being solved as well as a description of how these equations are formulated for numerical solution. Paragraph 2 gives the history and a perspective of the numerical algorithms that were developed during the course of this research. Paragraph 3 summarizes some of the accomplishments and results. Section III covers the research carried out on grid generation. Section IV gives recommendations for future work.

## SECTION II FLOW SOLVERS

### 1. EQUATION FORMULATION

The equations used in this research were the three-dimensional unsteady compressible Euler and Navier-Stokes equations. In some cases both Euler and Navier-Stokes equations were solved, with the Navier-Stokes equations being solved in viscous regions and the Euler equations solved elsewhere. This approach of solving the Navier-Stokes equations only in viscous regions does save some CPU time, but the savings are not large because it only takes 10-20 percent more CPU time to solve the Navier-Stokes equations per grid cell per time step than it does to solve the Euler equations per grid cell per time step (Reference 4). The largest cost in using Navier-Stokes equations is attributable to: (1) the increase in the number of cells required to resolve viscous regions, (2) the increase in the number of iterations sometimes required to converge the steady state Navier-Stokes solutions which is probably due to highly stretched grids, and (3) the reduction in the time step size for unsteady Navier-Stokes solutions which is due to the small grid cell size required to resolve viscous regions.

The three-dimensional compressible unsteady integral conservation law form of the flow equations was transformed from the Cartesian coordinate system to a time varying curvilinear computational domain (References 1,3, and 7). Coriolis terms, apparent mass terms, etc., were then captured naturally as part of the solution and not handled explicitly.

The integral conservation law form of the equations was solved in a cell centered finite volume formulation (Reference 1) in order to handle arbitrary geometries and eliminate certain conservation difficulties associated with the finite difference formulation. Also, it was pointed out in Reference 46 that the finite volume formulation yields superior results compared to the finite difference formulation for solution adaptive problems, which was part of this research. Perhaps the most compelling reason for using the finite volume formulation was that upwind schemes were used to solve the flow equations and it is straight forward

to maintain conservation in the finite volume formulation, whereas, it is most difficult (if it can be done at all) to maintain conservation in the finite difference formulation of upwind schemes for steady or unsteady flow solutions about arbitrary geometries.

In summary, the equations used were the three-dimensional compressible unsteady Euler and Navier-Stokes equations. These equations were transformed to a time dependent curvilinear coordinate system and discretized in a cell centered finite volume formulation for numerical solution.

## 2. NUMERICAL ALGORITHMS INVESTIGATED: A HISTORY AND PERSPECTIVE OF SKOAL, COPEN, BMULE, AND REDOX

During the course of this research four basic algorithms were developed for the numerical solution of the Euler and Navier-Stokes equations. Each algorithm was thought to be an improvement over the preceding one. All of the algorithms were finite volume upwind schemes of one type or another. For example, the first three algorithms, SKOAL, COPEN, and BMULE, were based on flux vector splitting and the last, REDOX, was based on flux difference splitting. The first two algorithms, SKOAL and COPEN, were explicit, and the remaining two algorithms were implicit. The first three algorithms were first or second order accurate in space or time. The fourth algorithm, REDOX, was first or second order accurate in time, and first, second, or third order accurate in space. Flux limiters were not used in the first three algorithms, as the flow regime of interest was transonic and limiters were never needed. However, the fourth algorithm, REDOX, was a high resolution scheme, and three different limiters were used. Each algorithm is briefly described in the chronological order in which it was developed, and the reasons are given why each algorithm was thought to be an improvement over the preceding one.

a. SKOAL

SKOAL is a three-dimensional unsteady finite volume flux vector split explicit code (References 1 and 14). According to Van Leer (Reference 47) it was the first three-dimensional flux vector split code ever written. A feature unique to the SKOAL code and not used in any of the succeeding codes, was the way the eigenvalues were evaluated at the cell faces. The eigenvalues at the cell faces were evaluated by averaging the dependent variables at the cell centers located on either side of the cell face. These cell face eigenvalues were used to determine the direction of propagation of information across the cell face, and then each piece of the split flux vector was appropriately upwind differenced. Once the direction of information passage was established, the eigenvalues were not recomputed using one sided differences, rather the eigenvalues as determined by the averaging of the dependent variables were used. This caused some smearing of the solution such as smearing shocks over four or five cells. As a point of interest, much sharper solutions could be obtained by reevaluating the eigenvalues using one sided differences based on the sign of the eigenvalues obtained by the dependent variable averaging. However, this led to instabilities in regions where the sign of the eigenvalues obtained by the two different methods differed. Consequently, the eigenvalues used were those obtained by averaging the dependent variables.

The CFL limit of the explicit scheme used was two. The SKOAL code was reasonably robust and numerous computations about rather complex configurations were computed such as those reported in References 1 and 14, and in particular those reported by Lijewski (References 23 and 24).

b. COPEN

The only difference between the COPEN and SKOAL codes was the way the eigenvalues were evaluated at cell faces. Rather than average the dependent variables on either side of a cell face as was done in SKOAL, the dependent variables were extrapolated to the cell face from either

side. Therefore, each cell face had associated with it a set of eigenvalues on the left side of the cell face and a set of eigenvalues on the right side of the cell face, and corresponding to each side of the cell face was a set of left and right fluxes. The flux at a cell face was then obtained by a linear combination of the fluxes on each side of the cell face (Reference 12). The COPEN code was also used for several computations, including a store moving away from a second body (Reference 12).

c. BMULE

The BMULE code is simply an implicit version of the COPEN code. The particular implicit scheme used is a block triangular scheme and is described in References 3, 29, and 38. This scheme requires only two passes through the computational domain at each time step rather than the three passes that are required by the more frequently used block tridiagonal type scheme of Beam and Warming (Reference 48). Also, the operation count is less for each pass of the block triangular scheme than it is for each pass of the block tridiagonal scheme. In addition, it has more favorable stability properties than the block tridiagonal scheme as pointed out by Anderson (Reference 49).

A disadvantage of the block triangular scheme is that it is a backward and forward substitution method and, as such, inherently resists vectorization. However, Belk and Janus have vectorized this portion of the code by passing through diagonal planes. An explanation of this vectorization process is given in References 3 and 29. With the vectorization of the block triangular solution scheme, the code was fully vectorized.

It turned out that the BMULE code was faster per time step per grid point than either of the preceding explicit schemes on a Cray X-MP. The reason for this was that the explicit codes required three flux balances per time step, whereas, the implicit code required only one. The implicit solution process does take longer than the explicit solution process, but this is outweighed by the reduced number of flux balances required. Since the implicit scheme requires fewer time steps to reach

convergence (typically about one-fifth the number of time steps to get to the same level of convergence) than does the explicit scheme for steady state problems, the end result is a large saving in CPU time. The implicit scheme does require much larger memory than does the explicit scheme, but efficient use of the solid state device (SSD) on the Cray X-MP minimizes the additional memory cost.

#### a. REDOX

The REDOX code is the same as the BMULE code except for the method used to perform the flux balances for the right hand side of the equation. Flux vector splitting was used to perform the flux balances in the BMULE code, whereas, flux differencing splitting based on Roe averaging was used in the REDOX code. The REDOX code is a three-dimensional, steady or unsteady, finite volume, implicit, upwind, high resolution (TVD), flux difference split scheme that can be run first or second order in time and first, second, or third order in space on stationary or dynamic, blocked or unblocked, grids for solving the Euler or Navier-Stokes equation in subsonic, transonic, supersonic, or hypersonic flow. Three flux limiters, minmod, Superbee, and Van Leer, are available in the code. The best descriptions of the CFD technology in the REDOX code are given in References 7, 39, and 50.

The original intent in constructing the REDOX code was to linearize the equations as is normally done in developing an implicit scheme. However, several problems were encountered in this approach having to do with the formal difficulty of linearizing terms involving Roe variables. For example, the scheme obtained from making certain linearization approximations of the Roe variables experienced convergence difficulties. It was found that if the left hand side of the BMULE code was used (that is, the BMULE solution matrix), then convergence was much improved. This means that flux vector splitting is used on the left hand side and flux difference splitting on the right hand side. It is the flux differencing splitting on the right hand side that controls the quality of the solution.

A disadvantage of the REDOX code compared to the BMULE code is that it takes 20 to 40 percent more CPU time per time step per grid point depending on the problem. However, an advantage of the REDOX code is that shocks and contact discontinuities are captured much more sharply. In addition, viscous regions can be accurately resolved with far fewer points. Numerical solutions from the BMULE and REDOX codes are compared with the Blasius solution for a simple flat plate laminar boundary layer in Reference 7. In this case, the REDOX thin-layer Navier-Stokes solution is in good agreement with the Blasius solution with only three points in the boundary layer, whereas, neither the coarse or fine grid BMULE solutions agree very well with the Blasius solution. The BMULE code can be made to agree with the Blasius solution but many points are required. This example required about the same number of cycles to converge both codes, so the increase in CPU time per point is made up for by the possibility of using fewer grid points.

The REDOX code is presently considered to be the best and most powerful code of those developed thus far. It is also the only one that is under further development, and it presently receives 90 percent of the total effort devoted to maintenance.

### 3. RESULTS

#### a. Innovations

Those things having to do with the numerical solution of the Euler and Navier-Stokes equations that can be considered innovations include:

- (1) SKOAL was the first three-dimensional unsteady completely upwind Euler code.
- (2) It was found for the REDOX code, which used flux differencing splitting on the right hand side of the equations, that the use of flux vector splitting in the solution matrix on the left hand side of the equations, led to a more stable, robust, and rapidly convergent scheme, than when flux differencing splitting was used in the solution matrix on the left hand side of the equations.

- (3) It is acceptable for the cases considered, to use Jacobian freezing when using Newton-type subiterations in order to save CPU time.
- (4) It is possible to accelerate Newton-type subiterations by the use of two time steps, one local and one minimum. The local time step is used much like in a steady state solution, where it appears as a coefficient of the residual on the right hand side of the equation. The residual, however, contains a time derivative in order to obtain true time accuracy with the minimum time step appearing in the time derivative.
- (5) It is possible to obtain time accurate numerical results for fine grid transonic Navier-Stokes solutions of an oscillating supercritical wing in turbulent flow, with maximum CFL numbers of the order of ten to the fourth power.
- (6) Accurate Navier-Stokes numerical solutions for laminar viscous flow over a flat plate can be obtained with only three grid cells in the viscous region using the high resolution REDOX scheme.

b. Application

To illustrate the type of results that can be obtained, numerical solutions are presented for the mutually interfering transonic flow about a multi-body wing-pylon-store configuration. The solution for this configuration was the basic objective of this research effort. This example illustrates the capability of the code to compute both steady state multi-block solutions and unsteady dynamic multi-block solutions for flow about complex configurations. Steady state multi-block solutions are demonstrated by computing the flow about the wing-pylon-store configuration with the store in the captive position. Unsteady dynamic multi-block solutions are demonstrated by computing the flow about the complete multi-body configuration as the store moves away from the parent wing-pylon configuration through a vertical launch trajectory. Unfortunately, no experimental data are available for comparison with the unsteady moving store solution; however, experimental data are available for the captive position and are compared with the numerical solutions.

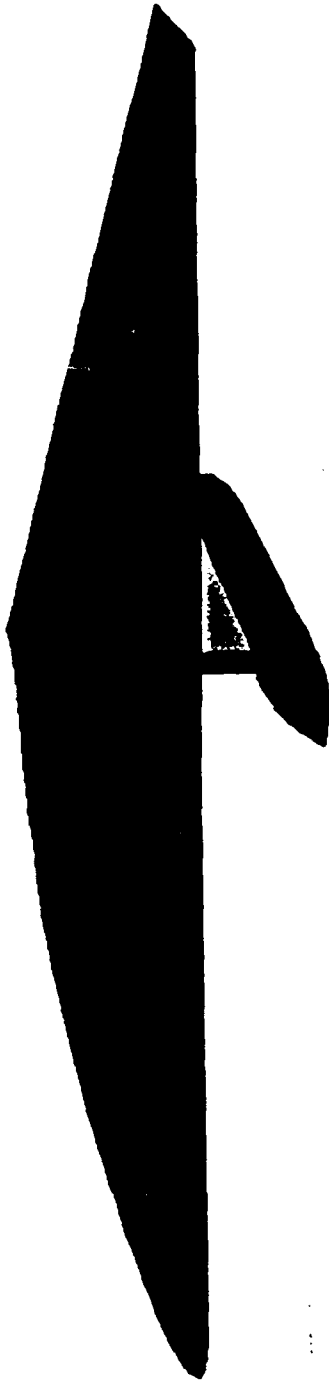
Grid generation for this configuration is discussed in Section III of this report. The wing-pylon-store configuration considered was the same as that used in wind tunnel experiments. The basic configuration

is shown in Figure 1-a with the store in the captive position and in Figure 1-b with the store located two store diameters below the pylon. The wing was a symmetrical airfoil and the leading edge was swept 45 degrees. The store was an ogive-cylinder with a cylindrical sting joined to the store boattail. The pylon was a biconvex airfoil shape, and a small gap existed between the store and pylon in both the experimental and computational configuration. The complete grid was composed of 30 blocks.

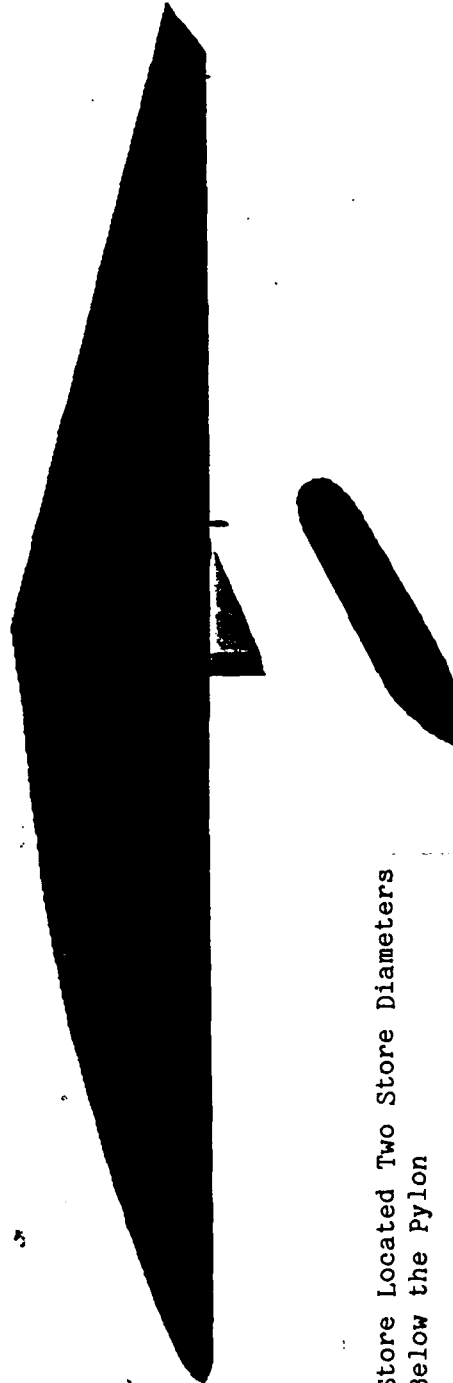
The numerical solution was run for a freestream Mach number of 0.85 and zero degrees angle of attack. Numerical and experimental surface pressure distributions on the outboard and inboard sides of the store (see Figure 2) in the captive position are shown in Figures 3 and 4, respectively. Notice that there is a large lower pressure region on the inboard side of the store (Figure 4) than on the outboard side of the store (Figure 3). Figures 5 and 6 are included to show that the same thing happens, computationally and experimentally, on the pylon. The result of this pressure differential would be that a released store would have an initial side force that would push the store toward the fuselage rather than away from the fuselage.

The reason for the pressure being lower on the inboard side of the store and pylon is attributed primarily to the presence of the store. Figures 7 and 8 are used to argue this point. These figures compare computations corresponding to the store dropping through a point 2 store diameters away from the pylon with steady state experimental data for the wing and pylon only (no store) at the same flow conditions. Notice that the inboard and outboard pressures on the pylon without the store present are now much closer to the same values. (One should note that it is dangerous to compare unsteady computations with steady state experimental data, but unsteady experimental data are not available and the assumption is made that the store being 2 diameters away will not significantly influence the unsteady flow about the wing and pylon.)

The only way to truly get an idea of the flowfield about such a configuration is through color graphics. Figure 9 shows the surface pressure distribution from a view above the wing-ylon-store configuration



a. Store in Captive Position



b. Store Located Two Store Diameters  
Below the Pylon

Figure 1. Wing-Pylon-Store Configuration

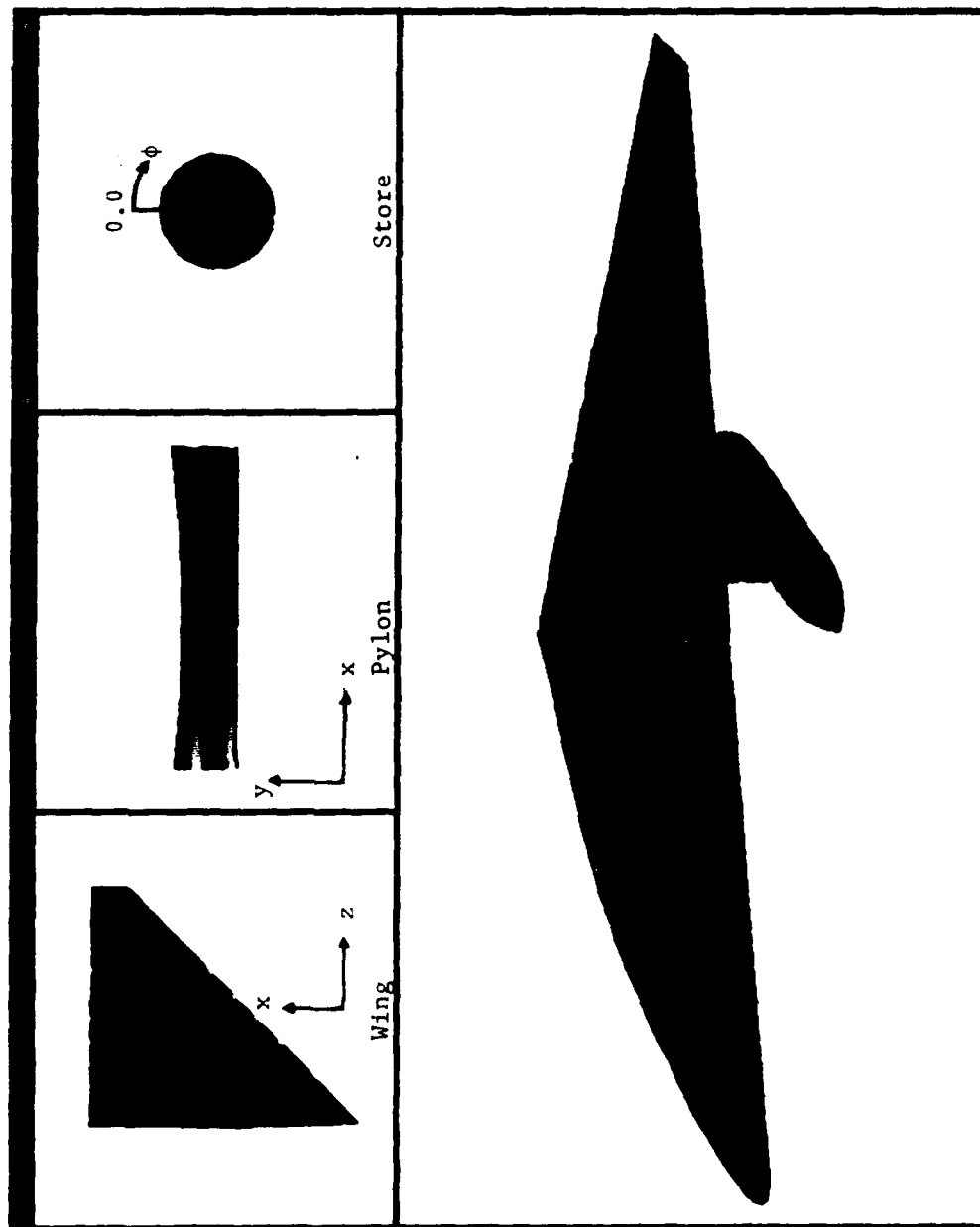


Figure 2. Wing-Pylon-Store Configuration Showing Locations of Numerical and Experimental Comparisons

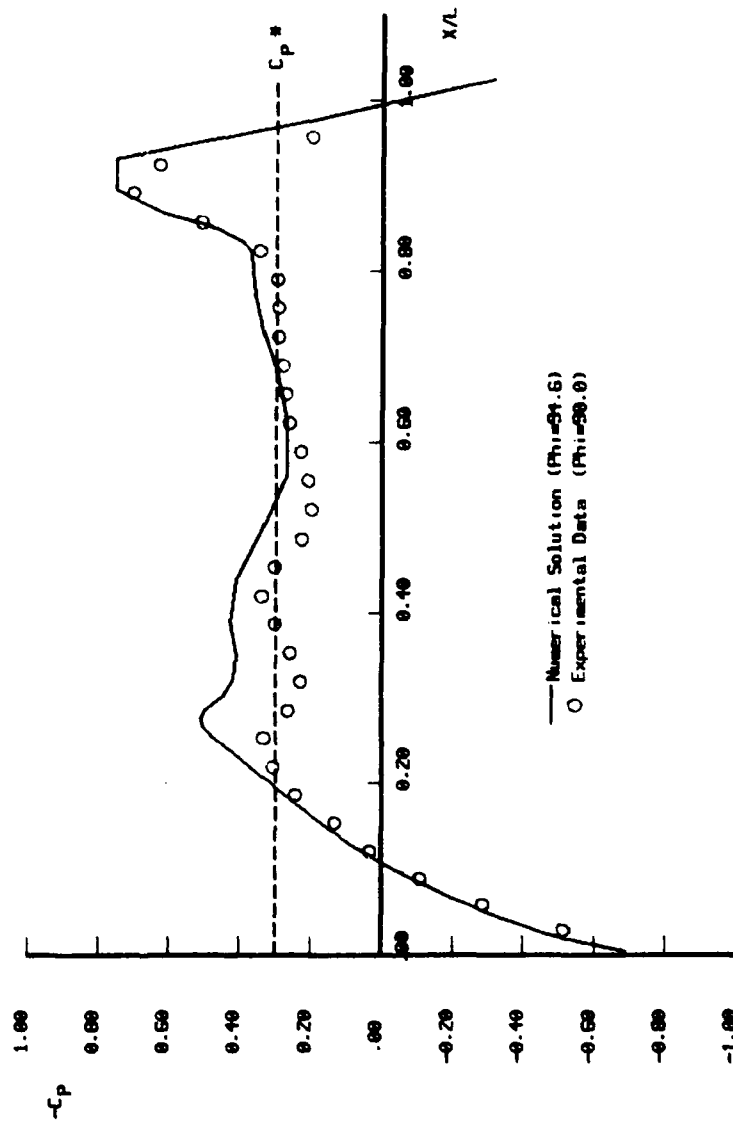


Figure 3. Numerical and Experimental Surface Pressure Distributions on the Outboard Side of the Store at  $M=0.85$ ,  $\text{Alpha}=0.0$

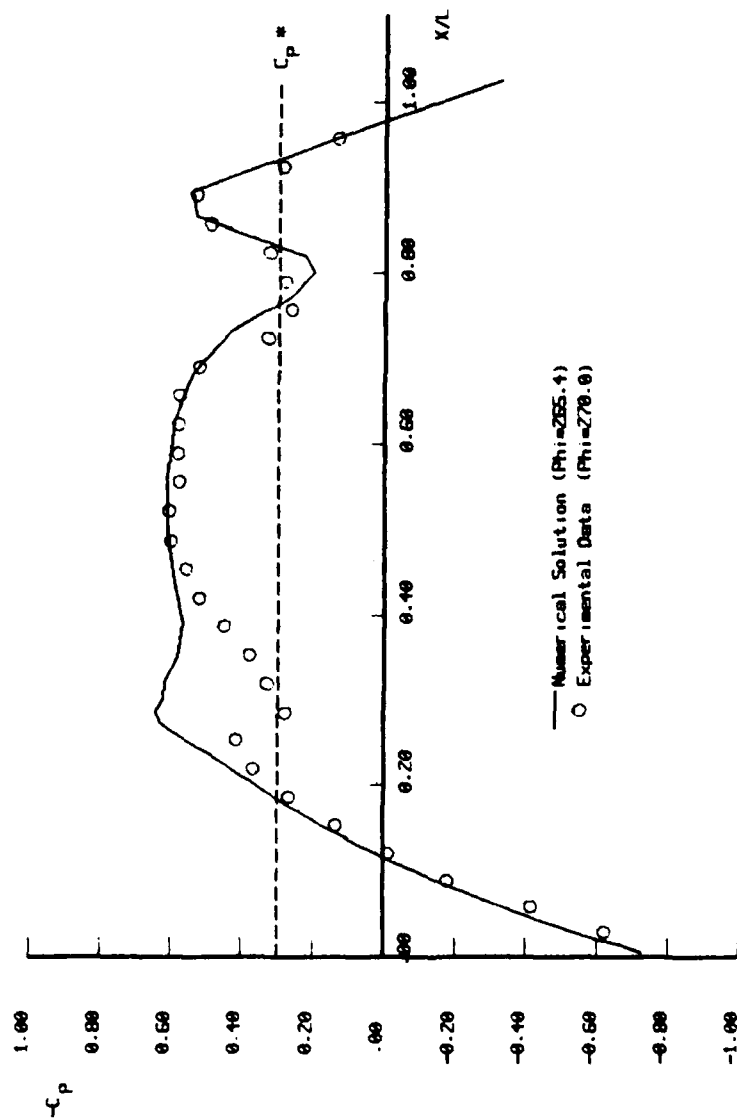


Figure 4. Numerical and Experimental Surface Pressure Distributions on the Inboard Side of the Store at  $M=0.85$ ,  $\text{Alpha}=0.0$

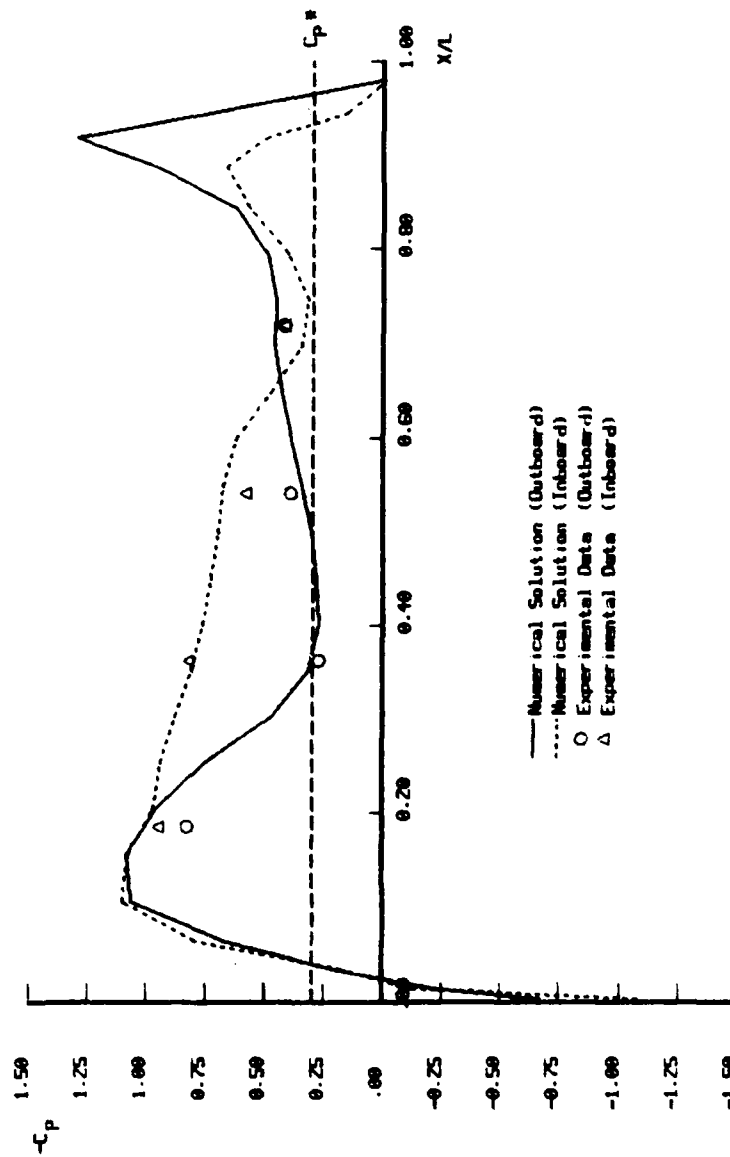


Figure 5. Numerical and Experimental Surface Pressure Distributions on the Pylon (Lower Row,  $Y=0.67$ ) at  $M=0.85$ ,  $\text{Alpha}=0.0$  (Captive Position)

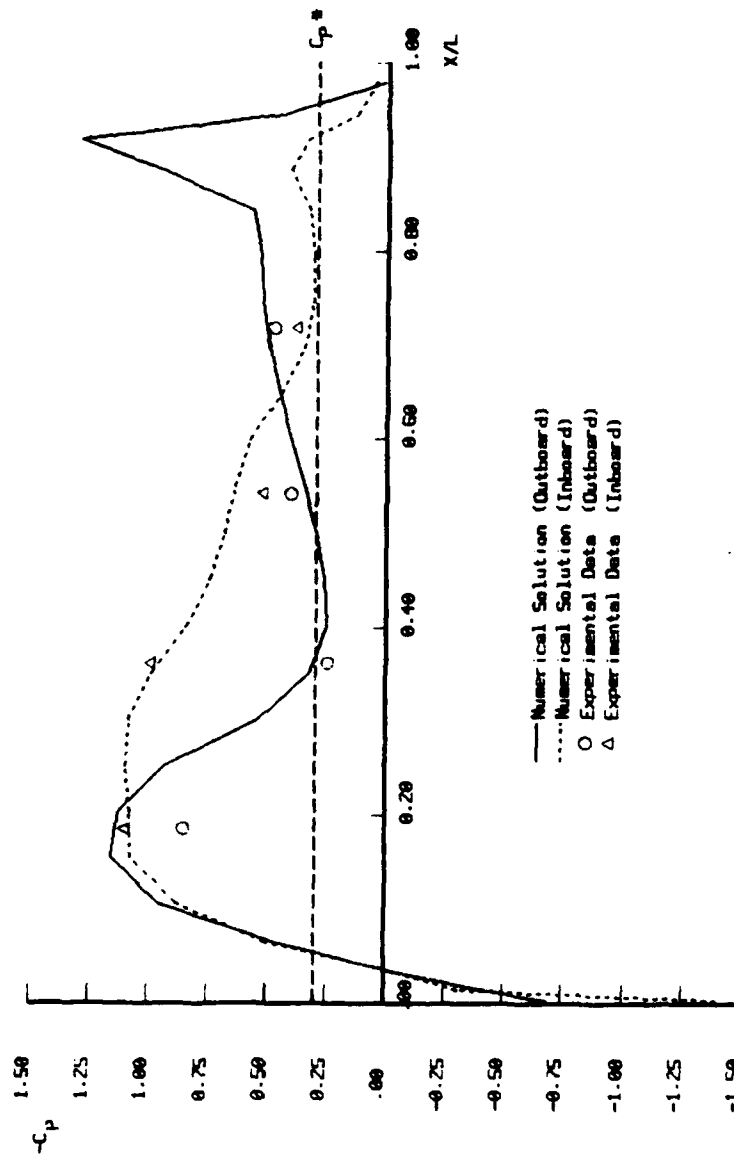


Figure 6. Numerical and Experimental Surface Pressure Distributions on the Pylon (Upper Row,  $Y=1.17$ ) at  $M=0.85$ ,  $\alpha=0.0$  (Captive Position)

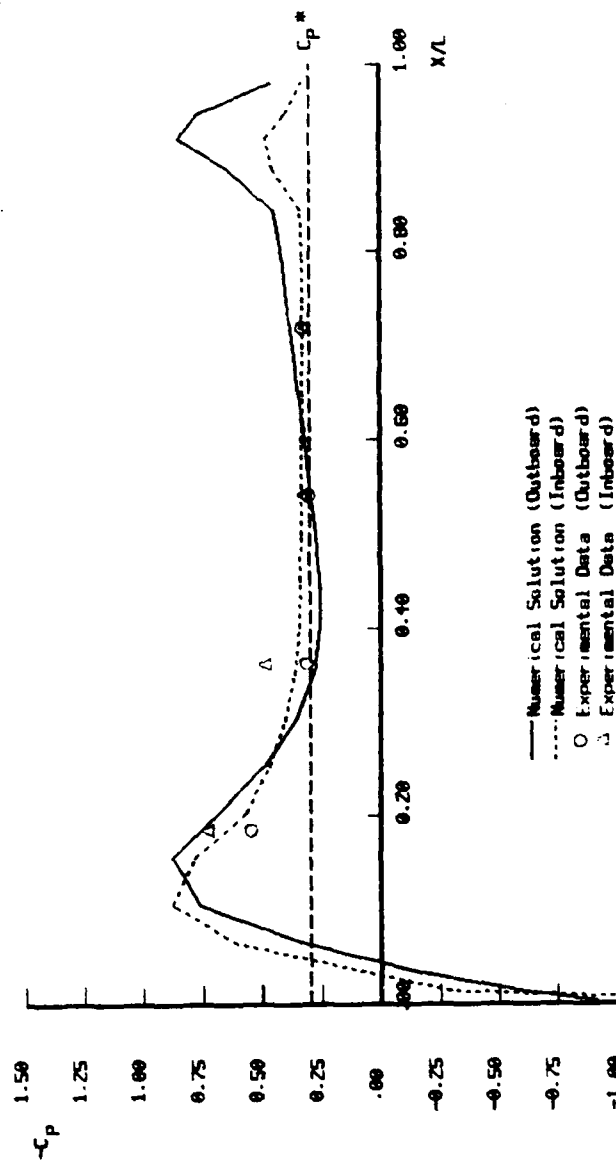


Figure 7. Numerical and Experimental Surface Pressure Distributions on the Pylon (Lower Row,  $Y=0.67$ ) at  $M=0.85$ ,  $\alpha=0.0$  (Released Position)

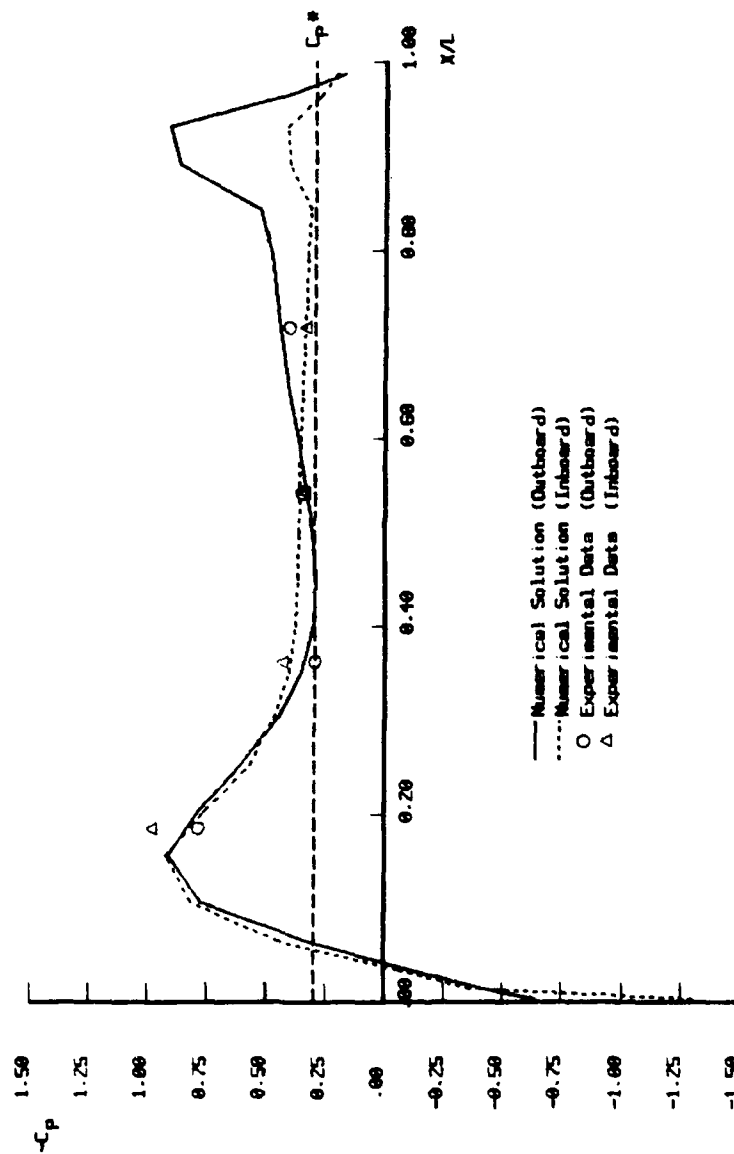
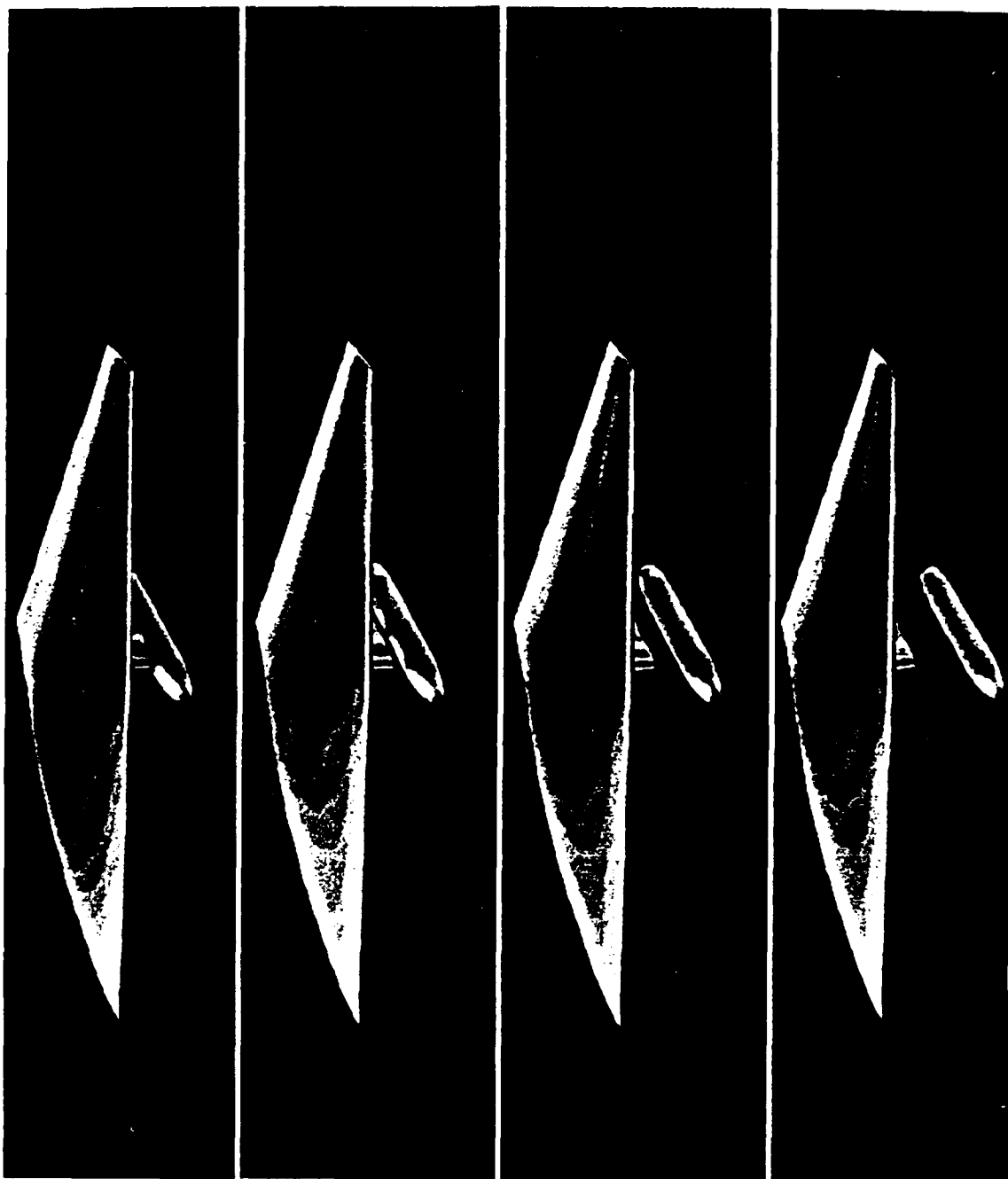


Figure 8. Numerical and Experimental Surface Pressure Distributions on the Pylon (Upper Row,  $Y=1.17$ ) at  $M=0.85$ ,  $\alpha=0.0$  (Released Position)

PRESSURE

psi/si



CONTOUR LEVELS

0.34000  
0.34000  
0.35500  
0.37000  
0.38500  
0.40000  
0.41500  
0.43000  
0.44500  
0.46000  
0.47500  
0.49000  
0.50500  
0.52000  
0.53500  
0.55000  
0.56500  
0.58000  
0.59500  
0.61000  
0.62500  
0.64000  
0.65500  
0.67000  
0.68500  
0.70000  
0.71500  
0.73000  
0.74500  
0.76000  
0.77500  
0.79000  
0.80500  
0.82000  
0.83500  
0.85000  
0.86500

Figure 9. Computed Surface Pressure Contours with Store Located in Captive Position and Moving Through 0.4, 1.0, and 2.0 Store Diameters Below the Pylon - View 1

as the store moves away from the wing and pylon. The top insert in Figure 9 is for the store in the captive position at the instant of vertical launch. The other inserts on the figure are for when the store is moving through the points of 0.4, 1.0, and 2.0 store diameters away from the pylon. Note that even on the upper surface the pressure changes, particularly when the store is near the pylon. Figure 10 shows the same sequence of positions from a similar view below the wing. Figure 11 is another view from below the wing showing the same sequence, but it is a view that makes it easier to see more detail. In these figures, red is the highest pressure, followed in order by yellow, green, and blue, where blue is the lowest pressure shown. The flow about the store is not influenced significantly by the presence of the wing-pylon by the time the store passes through a point 2 store diameters away from the pylon.

This 30-block wing-pylon-store example was not a trivial case, either from the grid generation point of view, or the flow solution point of view. However, once a grid is constructed and a flow solution obtained, any modifications to the configuration or flow conditions can be carried out rather rapidly, and new solutions obtained.

One of the primary difficulties associated with this work is post processing of the results, particularly for the unsteady computations. For unsteady computations the time required to transmit the necessary grid and flowfield information is prohibitive, even on high speed data lines. It is suggested that a way around this problem may be to download a restart file from the Cray computer (a restart to a solution that has been essentially completed on a Cray) to a large workstation (that is, one with lots of in-core memory, disk space, high quality graphics, and is capable of sustaining 10 or more mflops), and then running it on the workstation (albeit for a long period of time) and using the high quality workstation graphics for post processing a few time steps. Now having said this, it must be pointed out that this approach will not work effectively for the store separation problem just addressed if it is desired to make an animation composed of every time step of the computation of the store trajectory. However, for unsteady problems that become periodic, or for those problems where it is not necessary to

PHOTO

# CONTOUR LEVELS

0.25000  
0.30000  
0.34000  
0.35500  
0.37000  
0.38500  
0.40000  
0.41500  
0.43000  
0.44500  
0.46000  
0.47500  
0.49000  
0.50500  
0.52000  
0.53500  
0.55000  
0.56500  
0.58000  
0.59500  
0.61000  
0.62500  
0.64000  
0.65500  
0.67000  
0.68500  
0.70000  
0.71500  
0.73000  
0.74500  
0.76000  
0.77500  
0.79000  
0.80500  
0.82000  
0.83500  
0.85000  
0.86500

PHOTO

## PRESSURE

PHOTO

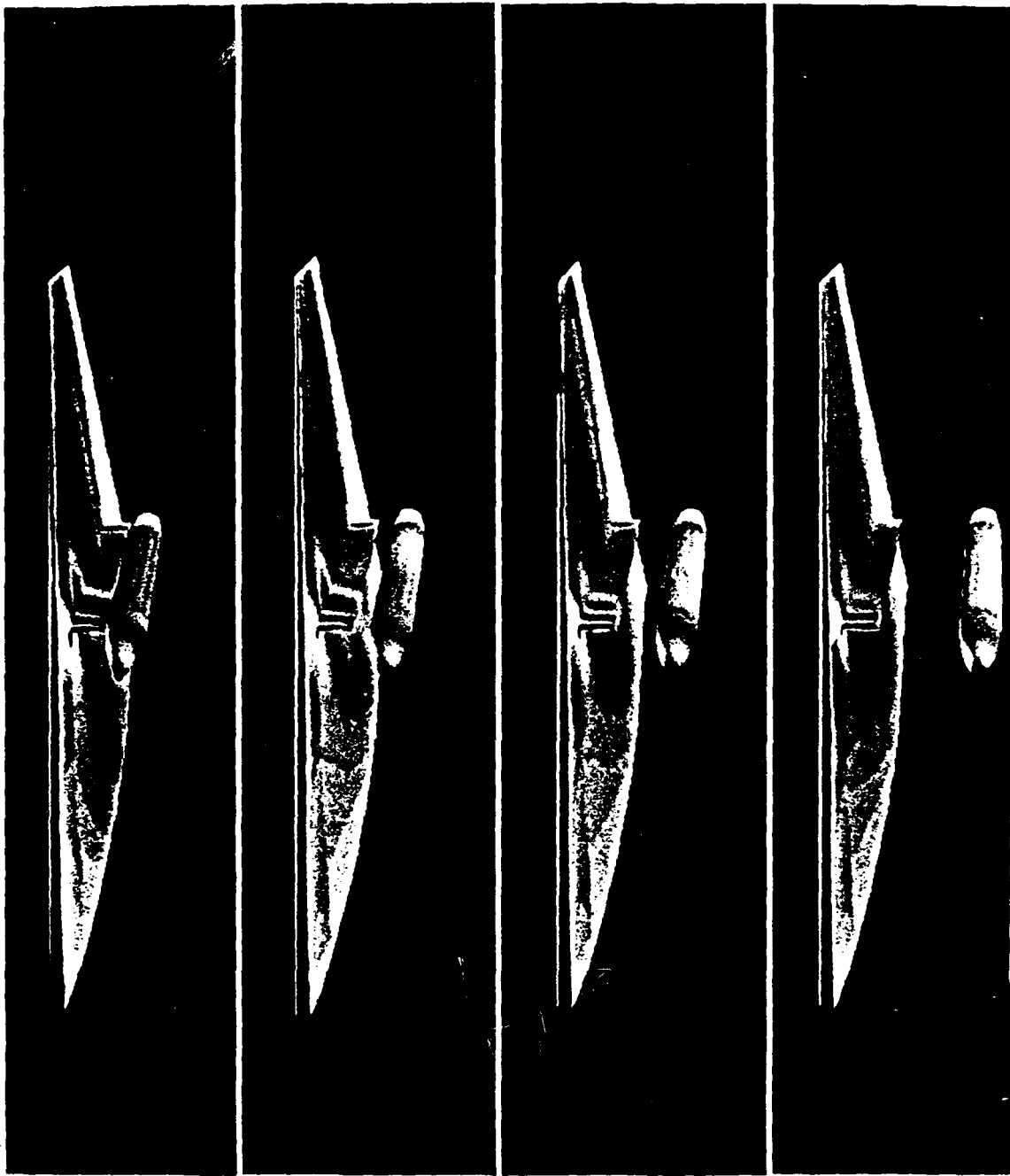


Figure 10. Computed Surface Pressure Contours with Store Located in Captivity Position and Moving Through 0.4, 1.0, and 2.0 Store Diameters Below the Pylon - View 2

photo

# PRESSURE

photo

photo

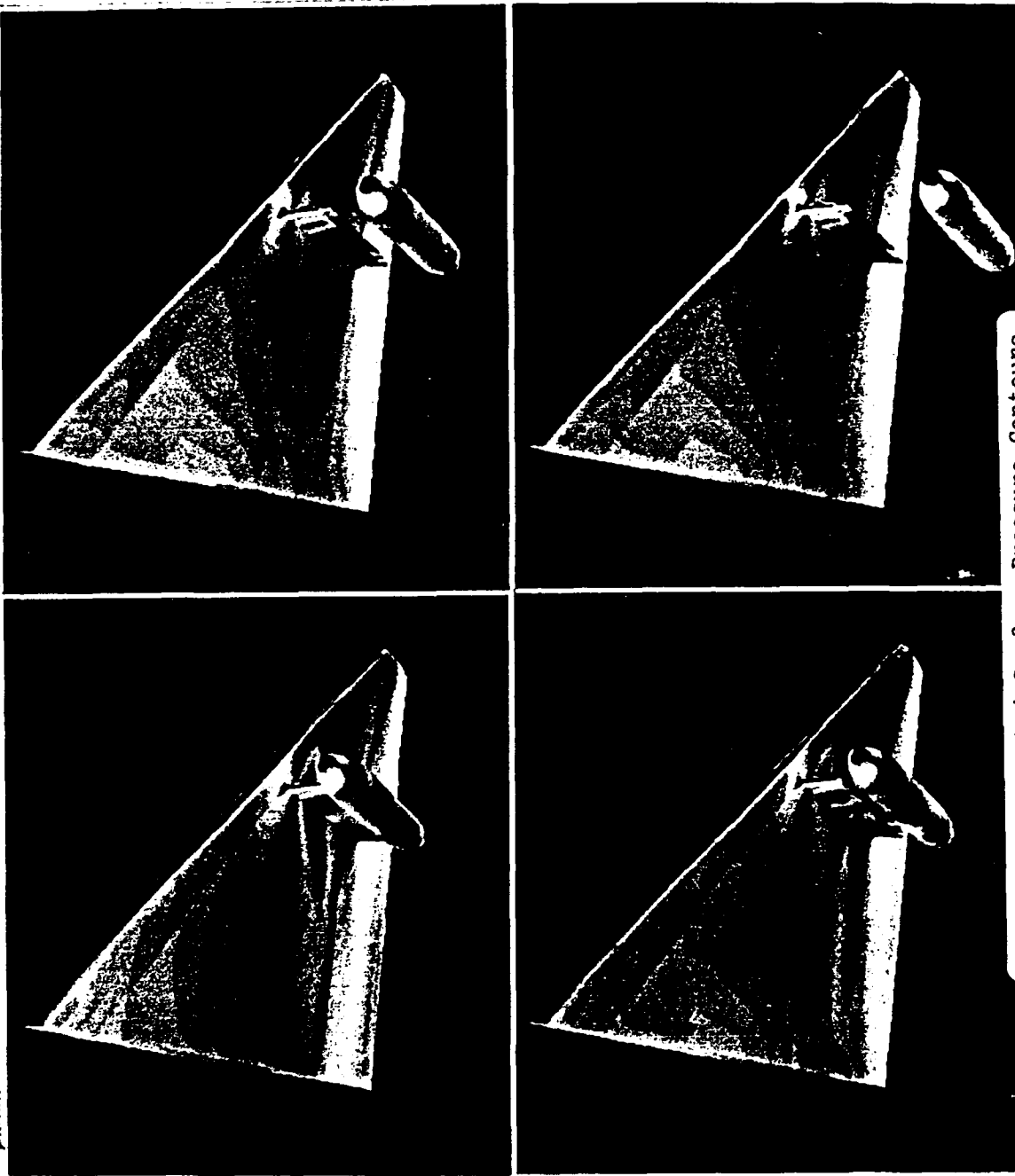


Figure 11. Computed Surface Pressure Contours with Store Located in Captive Position and Moving Through 0.4, 1.0, and 2.0 Store Diameters Below the Pylon - View 3

## CONTOUR LEVELS

0.32500  
0.34000  
0.35500  
0.37000  
0.38500  
0.40000  
0.41500  
0.43000  
0.44500  
0.46000  
0.47500  
0.49000  
0.50500  
0.52000  
0.53500  
0.55000  
0.56500  
0.58000  
0.59500  
0.61000  
0.62500  
0.64000  
0.65500  
0.67000  
0.68500  
0.70000  
0.71500  
0.73000  
0.74500  
0.76000  
0.77500  
0.79000  
0.80500  
0.82000  
0.83500  
0.85000  
0.86500

process every time step of the solution, this would seem to be a reasonable approach, at the present time, to post processing enormous amounts of information.

#### 4. CONCLUSIONS

Four numerical algorithms were developed during the course of this research for solving the unsteady Euler or Navier-Stokes equations for arbitrary three-dimensional geometries. Each algorithm was an improvement in computational technology over the preceding one. The last algorithm developed, REDOX, used flux vector splitting on the left hand side of the equation in the solution matrix, and flux difference splitting on the right hand side of the equation to compute the residual vector. The result was a high resolution upwind scheme for solving steady state or unsteady, inviscid or viscous flows, about arbitrary configurations. The numerical results were validated by comparisons with available experimental wind tunnel data for a wing-pylon-store configuration with the store in the captive and vertical launch positions.

### SECTION III

#### GRID GENERATION SYSTEM

##### 1. GENERATION SYSTEM FORMULATION

In order to computationally treat general multi-store configurations including fuselage, wing, and pylon, the development of a general, three-dimensional grid generation system was an essential part of this project. The complexity of this geometrical configuration required a grid code of such generality that the resulting EAGLE grid code is applicable to arbitrary three-dimensional configurations, and thus has become a major resource to computational fluid dynamics and all other areas of computational field problems. This code has now been acquired by well over 100 governmental, industrial, and university agencies across the country.

The grid code is discussed in general in the sections that follow. Complete detail and user instructions are given in References 18 and 19, and some features are discussed in other references cited herein.

The formulation of the EAGLE grid code is discussed in Reference 27, and in complete detail in Reference 19, with an introductory discussion given in Reference 18. A brief summary follows here.

boundary segment and setting these values in the array of position vectors with one index constant. With values set on the sides of the rectangular array of position vectors in this manner, the generation of the grid is accomplished by determining the values of  $\zeta_{ijk}$  in the interior of the rectangular array from the specified boundary values on its sides, e.g., by interpolation or a PDE solution. The set of values,  $\zeta_{ijk}$ , then forms the nodes of a curvilinear coordinate system filling the physical region. A physical region bounded by six generally curved sides can thus be considered to have been transformed to a rectangular computational region on which the curvilinear coordinates are the independent variables.

#### a. Boundary-Conforming Structured Grids

Finite difference (or finite volume) and finite element solutions both require a discrete set of points or cells covering the physical field, and the efficiency of the computation is greatly enhanced if there is some organization to this set. This organization can be provided by having the discretization defined by the nodes of a curvilinear coordinate system filling the physical field. Such systems are readily available from handbooks for certain simple configurations such as regions that are cylindrical, spherical, elliptical, etc. For general regions of arbitrary shape, numerical grid generation provides the curvilinear system.

The curvilinear system can be constructed simply by setting values in a rectangular array of position vectors:

$$\underline{r}_{ijk} \quad (i=1,2,\dots,I; \quad j=1,2,\dots,J; \quad k=1,2,\dots,K)$$

and identifying the indices  $(i,j,k)$  with the three curvilinear coordinates. The position vector  $\underline{r}$  is a three-vector giving the values of the Cartesian coordinates  $(x,y,z)$  of a grid point. Since all increments in the curvilinear coordinates cancel out of the transformation relations for derivative operators, there is no loss of generality in defining the discretization to be on integer values of these coordinates.

Fundamental to this curvilinear coordinate system is the coincidence of some coordinate surface with each segment of boundary of the physical region, in the same manner that surfaces of constant radius coincide with the inner and outer boundary segments of the region between two concentric spheres filled with a polar coordinate system. This is accomplished by placing a two-dimensional array of points on a physical

## b. Composite-Block Structure

Although in principle it is possible to establish a correspondence between any physical region and a single empty rectangular block for general three-dimensional configurations, the resulting grid is likely to be much too skewed and irregular to be usable when the boundary geometry is complicated. A better approach with complicated physical boundaries is to segment the physical region into contiguous sub-regions, each bounded by six curved sides (four in 2D) and each of which transforms to a rectangular block in the computational region, with a grid generated within each sub-region (References 42,13,16). Each sub-region has its own curvilinear coordinate system irrespective of that in the adjacent sub-regions.

This then allows both the grid generation, and numerical solutions on the grid to be constructed to operate in a rectangular computational region, regardless of the shape or complexity of the full physical region. The full region is treated by performing the solution operation in all of the rectangular computational blocks. With the composite framework, partial differential equation solution procedures written to operate on rectangular regions can be incorporated into a code for general configurations in a straightforward manner, since the code only needs to treat a rectangular block. The entire physical field then can be treated in a loop over all the blocks.

The generally curved surfaces bounding the sub-regions in the physical region form internal interfaces across which information must be transferred, i.e., from the sides of one rectangular computational block to those of another. These interfaces occur in pairs, an interface on one block being paired with another on the same or different block, since both correspond to the same physical surface. Grid lines at the interfaces may meet with complete continuity, with or without slope continuity, or may not meet at all.

Complete continuity of grid lines across the interface requires that the interface be treated as a branch cut on which the generation system is solved just as it is in the interior of blocks. The interface locations are then not fixed, but are determined by the grid generation system. This is most easily handled in coding by providing an extra layer of points surrounding each block. Here the grid points on an interface of one block are coincident in physical space with those on another interface of the same or another block, and also the grid points on the surrounding layer outside the first interface are coincident with those just inside the other interface, and vice versa. This coincidence can be maintained during the course of an iterative solution of an elliptic generation system by setting the values on the surrounding layers equal to those at the corresponding interior points after each iteration. All the blocks are thus iterated to converge together, so that the entire composite grid is generated at once.

The construction of codes for complicated regions is greatly simplified by the composite grid structure since, with the use of the surrounding layer of points on each block, a flow code is only required basically to operate on a rectangular computational region. The necessary correspondence of points on the surrounding layers (image points) with interior points (object points) is set up by the grid code and made available to the computational fluid dynamics solution code.

The original impetus for using blocked grids was (1) to simplify the gridding of complex configurations, and (2) permit the solution of large problems requiring many grid points by keeping only the information needed to solve one block in central memory while retaining the information associated with the remaining blocks in secondary memory. Experi-

ence exposed a third reason for using blocked grids, having to do with computer cost and/or job turnaround. Supercomputer installations place a high price on the use of central memory, whereas, the use of secondary memory is relatively cheap. For example, when more than two million words of central memory are used on a Cray X-MP 2x4, the cost increases considerably on commercial machines, and the priority decreases considerably on government machines. By blocking, the use of central memory can be controlled, not only to fit the available memory, but also to fit the available budget and time constraints.

### c. Surface Grid System

The specification of the boundary point distribution is a two-dimensional grid problem in its own right, which can also be done either by interpolation or a Partial Differential Equations (PDE) solution. In general, this is a 2D boundary value problem on a curved surface, i.e., the determination of the locations of points on the surface from specified distributions of points on the four edges of the surface. This is best approached through the use of surface parametric coordinates, whereby the surface is specified by a 2D array of points,  $\zeta_{ij}$ , e.g. a set of cross-sections. The surface is then splined, and the spline coordinates (surface parametric coordinates) are then made the dependent variables for the interpolation or PDE generation system. The generation of the surface grid can then be accomplished by first specifying the boundary points in the array  $\zeta_{ijk}$  on the four edges of the surface grid, converting these Cartesian coordinate values to spline coordinate values on the edges, then determining the interior values of the spline coordinates from the edge values by interpolation or PDE solution, and finally converting these spline values to Cartesian coordinates.

The surface grid generation system is discussed in Reference 43, and in complete detail in References 51 and 19.

#### d. Adaptive Coupling

Finally, dynamically-adaptive grids continually adapt to follow developing gradients in the physical solution. This adaption can reduce the oscillations associated with inadequate resolution of large gradients, allowing sharper shocks and better representation of boundary layers. Another advantageous feature is the fact that in the viscous regions where real diffusion effects must not be swamped, the numerical dissipation from upwind biasing is reduced by the adaption. Dynamic adaption is at the frontier of numerical grid generation and may well prove to be one of its most important aspects, along with the treatment of real three-dimensional configurations through the composite grid structure.

With structured grids and implicit flow solvers, the adaptive strategy based on redistribution is by far the most simple to implement, requiring only the regeneration of the grid at each adaptive stage without modification of the flow solver unless time accuracy is desired. Time accuracy can be achieved, as far as the grid is concerned, by simply transforming the time derivatives, thus adding additional convective-like terms which do not alter the basic conservation form of the partial differential equations.

Two approaches for generating adaptive grids were investigated in the present study: (1) the control function approach based on the elliptic generation system, and (2) the variational approach based on the calculus of variations. Both methods have proven their capabilities for controlling the grids, but because of the longer computing time and less sensitivity in the variational method, the control function form should be the more promising tool for future applications. More detail and results are given in References 33, 5, and 8.

This adaptive control function formulation has been incorporated into the EAGLE grid code, providing a composite-block adaptive grid generation system for general three-dimensional regions to be coupled with PDE solvers.

## 2. CODE FOUNDATION

The development of the EAGLE grid generation system was founded on four essential items, the first of which was elliptic grid generation. The elliptic generation system is discussed in Reference 27, and in detail in References 44 and 19.

The second essential element of the foundation of the present code was the algebraic grid generation technique of transfinite interpolation developed in the automobile industry. The EAGLE code generates an algebraic grid by transfinite interpolation, which then serves as the initial solution to start the iterative solution for the smoother elliptic grid. (The algebraic grid can, of course, be taken as the final grid if desired). The algebraic generation system is discussed in Reference 27, and in complete detail in Reference 19.

The third component of the basis of the EAGLE code, and the item that is fundamental to the grid structure, was the technique of block-structured grids (Reference 42), which allows arbitrary configurations to be treated. This implementation of the block structure is discussed in Reference 27, and in complete detail in Reference 19.

Finally, a study was conducted as a part of this project to identify point distribution functions that reduce the truncation error induced into the CFD solution by the grid (Reference 10). These distribution functions were used in a number of places in the code, as is discussed in Reference 19.

### a. Design Considerations

A fundamental design criteria of the grid code was ease of use, and toward that end NAMELIST input was adopted even though not standard in FORTRAN 77. This input form was chosen because it allows the input to be read as English, and requires only the specification of essential items that differ from default values. This input form also creates a type of high-level language for grid generations. A NAMELIST emulator (Reference 2) was written in order to allow the code to be ported to systems not supporting NAMELIST.

Another feature that evolved from use of the code was the use of numbered points, curve and surface segments, and spacings corresponding to preliminary sketches (Reference 45). This was done in order to allow changes in spacings and in the number of points on segments to be made in a localized fashion without having to be repeated throughout the input runstream.

In order not to limit the size of configurations, provision was made for keeping only one block of the grid in core at a time, with the remainder stored on separate files on a solid-state disk (SSD) on the Cray X-MP, or on conventional disk files on other systems.

Provision was made for returning all or any number of parts of the grid for plotting, and a three-dimensional grid and contour plot code was written for the IRIS 2500 graphics system. (This graphics system was obtained from a DoD Research Equipment Grant for use on this project.) These plot codes have been ported to the IRIS 3010 and IRIS 4D/70 GT Systems.

As the code evolved it became clear that an error-checking mechanism was necessary to aid the user in the construction of complicated configurations, and therefore extensive error-checking was installed through which the code guides the user in the correction of input errors.

The total grid generation code package consists of a front-end boundary code (approximately 15,000 lines) which generates the boundary surface or curve segments for input to the grid code (about 20,000 lines). Both of these codes were written in modular form so that additional features can be easily incorporated, and new features have indeed continually been added. These codes are not static even now, but can continue to be extended in a straightforward manner to incorporate emerging new techniques in grid generation. The framework provided allows these codes to continue to evolve to include important features of other grid generation codes rather than having to compete with others.

## b. Grid System

The EAGLE grid code (Reference 19, Vol. III) is a general three-dimensional elliptic grid generation code based on the composite-block structure. This code allows any number of blocks to be used to fill an arbitrary three-dimensional region. Any block can be linked to any other block (or to itself) with complete (or lesser) continuity across the block interfaces as specified by input. In the case of complete continuity, the interface is a branch cut, and the code establishes a correspondence across the interface using a surrounding layer of points outside the blocks. This allows points on the interface to be treated just as all other points so that there is no loss of continuity. The physical location of the interface is thus totally unspecified in this case, being determined by the code.

This code uses an elliptic generation system with automatic evaluation of control functions either directly from the initial algebraic grid and then smoothed, or by interpolation from the boundary-point distributions. In the former case the smoothing is done only in the two directions other than that of the control function. This allows the relative spacing of the algebraic grid to be retained but on a smoother grid from the elliptic system. In the latter case, the arc length and curvature contributions to the control functions are evaluated and interpolated separately into the field from the appropriate boundaries. The control function at each point in the field is then formed by combining the interpolated elements. This procedure allows very general regions with widely varying boundary curvature to be treated.

The control functions can also be determined automatically to provide orthogonality at boundaries with specified normal spacing. Here the iterative adjustments in the control functions are made by increments radiated from boundary points where orthogonality has not yet been attained. This allows the basic control function structure evaluated from the algebraic grid, or from the boundary-point distributions, to be retained and thus relieves the iterative process from the need to establish this basic geometric form of the control functions.

Alternatively, boundary orthogonality can be achieved through Neumann boundary conditions which allow the boundary points to move over a surface spline, the boundary point locations being located by Newton iteration on the spline to be at the foot of normals to the adjacent field points. This is the boundary treatment used in the adaptive mode. Provision is also made for extrapolated zero-curvature boundary conditions and for mirror-image reflective boundary conditions on symmetry planes.

Although written for 3-D, the code can operate in a 2-D mode on either a plane or curved surface. In the case of a curved surface, the surface is splined and the generation is done in terms of surface parametric coordinates.

The code includes an algebraic three-dimensional generation system based on transfinite interpolation (using either Lagrange or Hermite interpolation) for the generation of an initial solution to start the iterative solution of the elliptic generation system. This feature also allows the code to be run as an algebraic generation system if desired. The interpolation, though defaulted to complete transfinite interpolation from all boundaries, can be restricted by input to any combination of directions or lesser degrees of interpolation, and the form (Lagrange, Hermite, or incomplete Hermite) can be different in different directions or in different blocks. The blending functions can be linear or, more appropriately, based on interpolated arc length from the boundary point distributions.

Blocks can be divided into sub-blocks for the purpose of generation of the algebraic grid and the control functions. Point distributions on the sides of the sub-blocks can either be specified or generated by transfinite interpolation from the edges of the side. This allows additional control over the grid in general configurations, and is particularly useful in cases where point distributions need to be specified in the interior of a block, or to prevent grid overlap in highly curved regions.

The composite structure is such that completely general configurations can be treated, the arrangement of the blocks being specified by input, without modification of the code. The input is user-oriented and

designed for brevity, easy recognition, and localized modification. For example, the establishment of correspondence, i.e., a branch cut, between two blocks requires only the simple input statement

```
$INPUT ITEM = "CUT", START = __, __, __, END = __, __, __, BLOCK = __,  
  
ISTART = __, __, __, IEND = __, __, __, IBLOCK = __$
```

where START and END give the three indices of two opposite corners of the cut section on one block (BLOCK), while ISTART and IEND give the corners of the corresponding section on the other block (IBLOCK). (The three indices can even be replaced by a single point number, corresponding to a previously-set numbered point.) The code sets up the point correspondence on the surrounding layers for complete continuity without additional input instructions.

The code is written in modular form so that components can be readily replaced. The code is vectorized (Cray X-MP) wherever practical and includes provision for separate storage of each block on the CRAY solid-state device (or conventional disk) to allow very large grids to be generated.

### c. Boundary System

An auxiliary front-end code (Reference 19, Vol. II) sets up boundary data for input to the grid code. This auxiliary code builds boundary segments in response to a series of input commands that again are designed to be user-oriented, brief, easily recognized, and localized. The following features are included: (1) generation of generic plane conic-section or cubic curves, (2) generation of cubic space curves, (3) generation of generic conic-section surfaces, (4) generation of cubic surfaces, (5) generation of surfaces by stacking, rotating, or blending curves, (6) extraction and concatenation of surface segments, (7) transformation of surfaces by translation, rotation, and scaling, (8) reversal or switching of point progressions on surfaces, (9) establishment of point distributions by curvature and with specified end, or interior,

spacings, (10) establishment of surface parametric grids by transfinite interpolation, (11) generation of tensor-product surfaces, (12) generation of surfaces by transfinite interpolation, (13) generation of grids on curved surfaces, and (14) intersection of surfaces.

This front-end code is coupled with the grid code through its output to allow changes to be strictly localized in the input runstream, i.e., changes in point distributions or numbers of points on a segment can be made at a single place in the boundary code runstream without having to be also made in the grid code runstream.

### 3. RESULTS

The EAGLE grid code has been applied to numerous configurations by researchers at Mississippi State and at Eglin AFB. Further applications have been made by other agencies that have acquired the code. Its development has contributed both a major computing resource to the CFD community and also some fundamental contributions to numerical grid generation.

#### a. Innovations

The development of the EAGLE grid code produced the following advances in numerical grid generation:

- (1) Improved automatic evaluation of control functions for elliptic generation systems from boundary point distributions (Reference 44). This involved the separate interpolation of curvature and arc length distribution effects, the use of transfinite interpolation for that interpolation, and a special interpolation based on hyperbolic tangent distribution functions to treat large variations in curvature.
- (2) A smoothing procedure for control functions that smoothes the functions in the two directions other than that in which the distribution is being controlled (Reference 44).
- (3) Evaluation of control functions from the algebraic grid, followed by smoothing (Reference 44). This retains the distribution from the algebraic grid but with a smoother grid.

- (4) Improved iterative adjustment of the control functions for boundary orthogonality (Reference 44). This involved the development of a procedure for radiating changes in the control functions into the field from all boundary points.
- (5) Automatic evaluation of locally optimum acceleration parameters for the SOR iteration in the elliptic generation system (Reference 44). A feedback stability limitation was also developed for use with iterative adjustment of control functions for boundary orthogonality.
- (6) Automatic adjustment to use directed differences for first derivatives, based on the sign of the control functions (Reference 44).
- (7) Newton iteration on a surface spline for the application of Neumann boundary conditions to achieve orthogonality or zero curvature at the boundary (Reference 27).
- (8) Full transfinite interpolation, based on either Lagrange or Hermite interpolation, for the algebraic grid (Reference 27). This involved the use of arc-length blending functions interpolated by transfinite interpolation of one less dimension, with automatic correction for zero arc lengths.
- (9) Full implementation of an elliptic surface grid generation system based on surface parametric coordinates in a composite-block structure with boundary orthogonality (Reference 43).
- (10) The identification of optimal distribution functions to reduce truncation error induced by the grid (Reference 10).

#### b. Applications

##### (1) General

Applications of the EAGLE grid code have been made to one, two, and three-store configurations. Some of these configurations have included pylon, wing, and fuselage. The stores have incorporated fins, wings, canards, and deflected fins. Results have been reported in References 6, 11, 15, 18, 24, 25, 26, 34, 40, 51 and other cited references.

The adaptively coupled grid and Euler codes have been applied to two and three-dimensional wings (References 5,33), and to single-store configurations (Ref. 8).

## (2) Wing-Pylon-Store Configuration

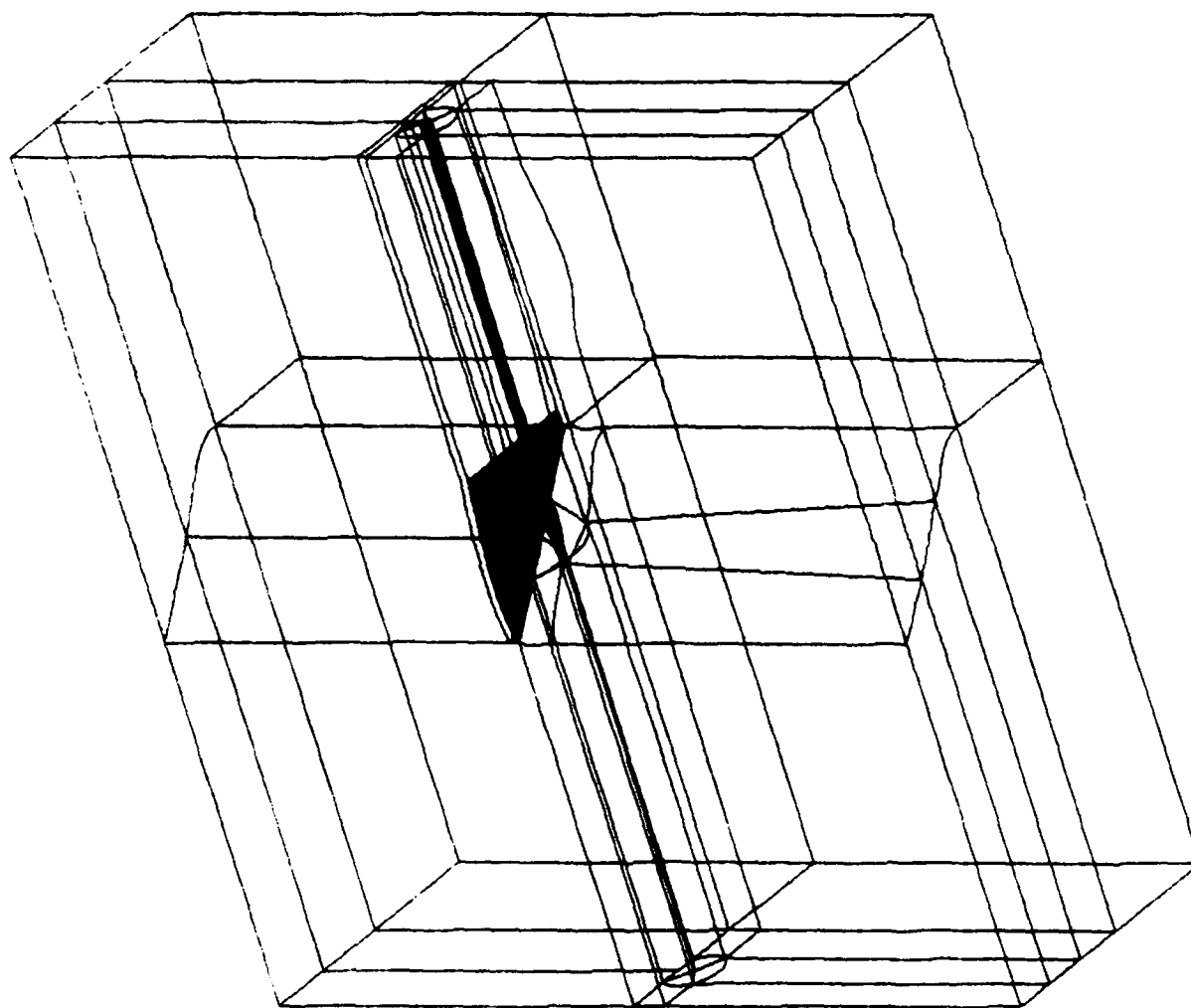
A particular application of the EAGLE grid generation system was the multi-body problem of the wing-pylon-store configuration in Figure 12. The wing is a symmetric wind tunnel model with 45 degrees of leading edge sweep. The pylon is a biconvex airfoil shape, while the store is an ogive-cylinder with an aft cylindrical sting.

The surface grid defining the hardware geometry was built entirely with operations in the EAGLE boundary code. Coordinates for the wing root and tip were read in, and the wing was built by interpolation between root and tip coordinates, using the BLEND operation. The chordwise point distribution was set by CURDIST, which allows control over spacing at both the leading and trailing edges.

The pylon, store, and sting were constructed according to dimensions and specifications by building up curve segments and then rotating these curves, or by interpolating between them with BLEND or TRANSUR. The pylon was generated separately and then affixed smoothly to the wing lower surface through operation INTSEC. Since the ultimate purpose of the grid was for modeling store separation through the unsteady Euler equations, a gap was left between the lower surface of the pylon and the top of the store, consistent with the wind tunnel model.

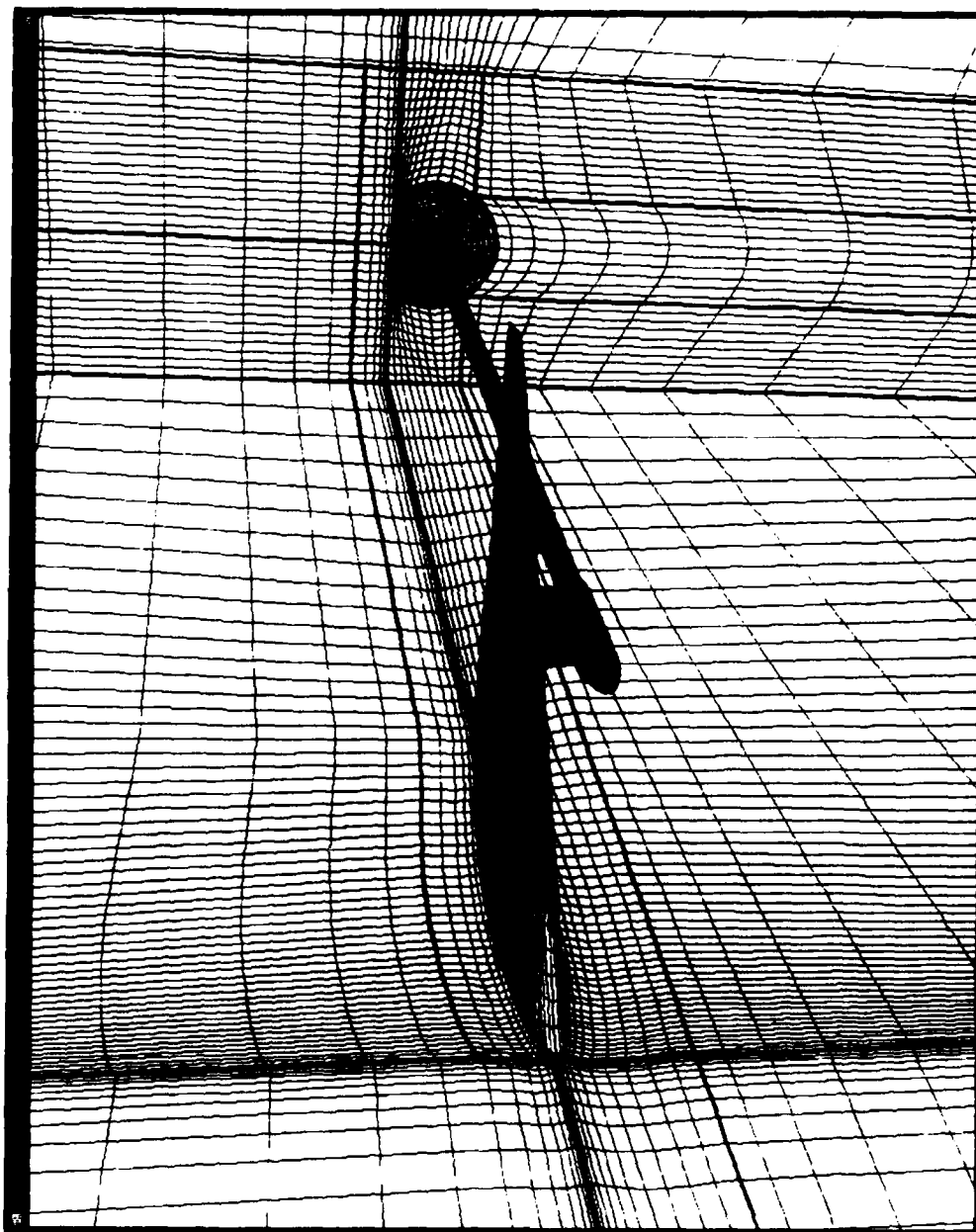
The grid developed for this problem is a 30-block system containing approximately 220,000 points. The block boundary curves are shown in Figure 12a. The system consists of a C-O grid enclosing the pylon, store, and sting, with an H-type grid surrounding the wing and the embedded C-O structure. An H-O structure extends from the nose of the C-O to the upstream grid boundary. The downstream and inboard boundary planes can be seen in Figure 12b.

The boundary for the C-O grid was generated by rotating curves about the axis of the store and sting. This approach, while quite simple for an isolated body, was complicated considerably by the need to have the C-O boundary interface smoothly with the highly curved lower surface of the wing. Additional complexity was caused by the 45-degree sweep angle of the wing leading edge.

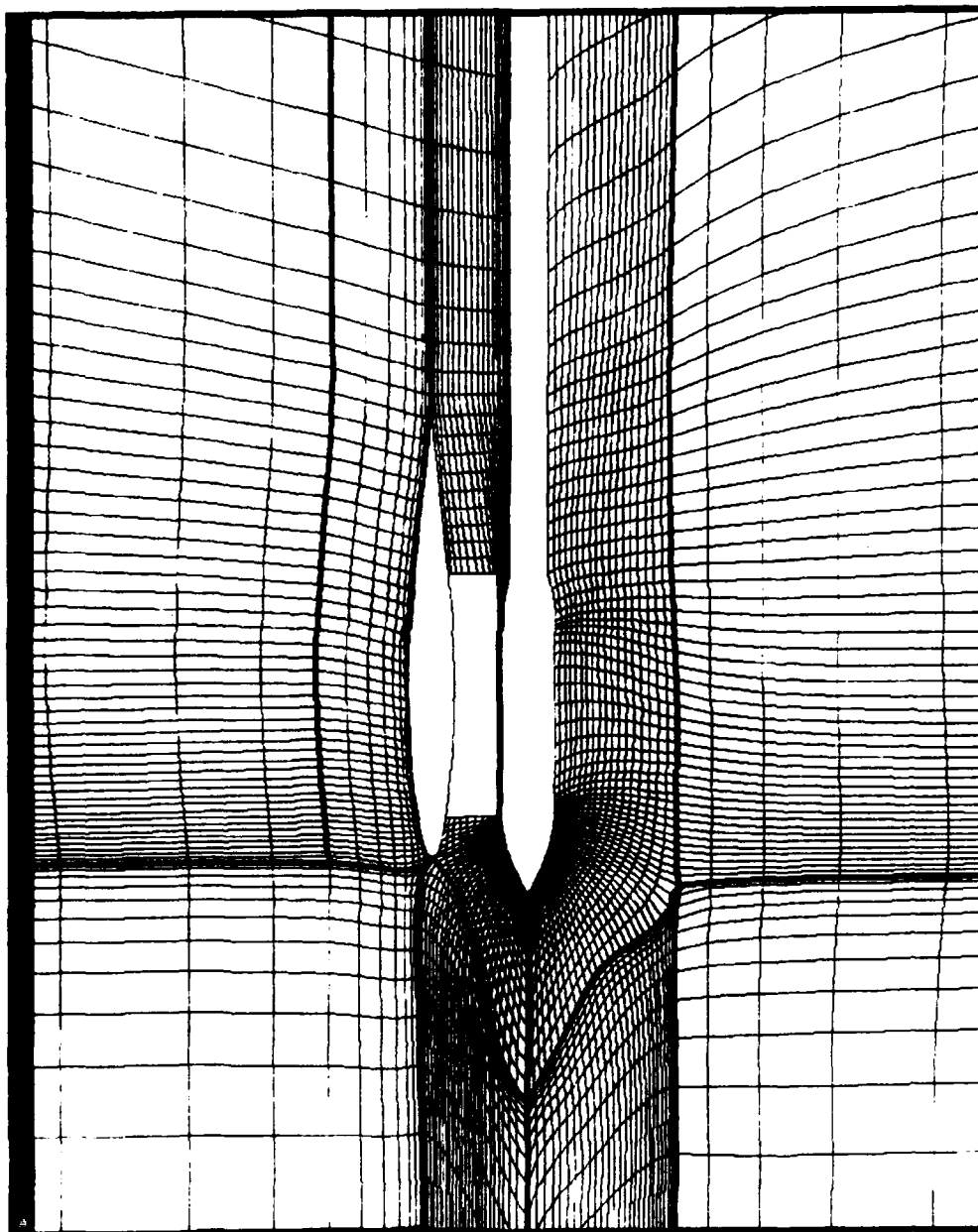


a. Perspective

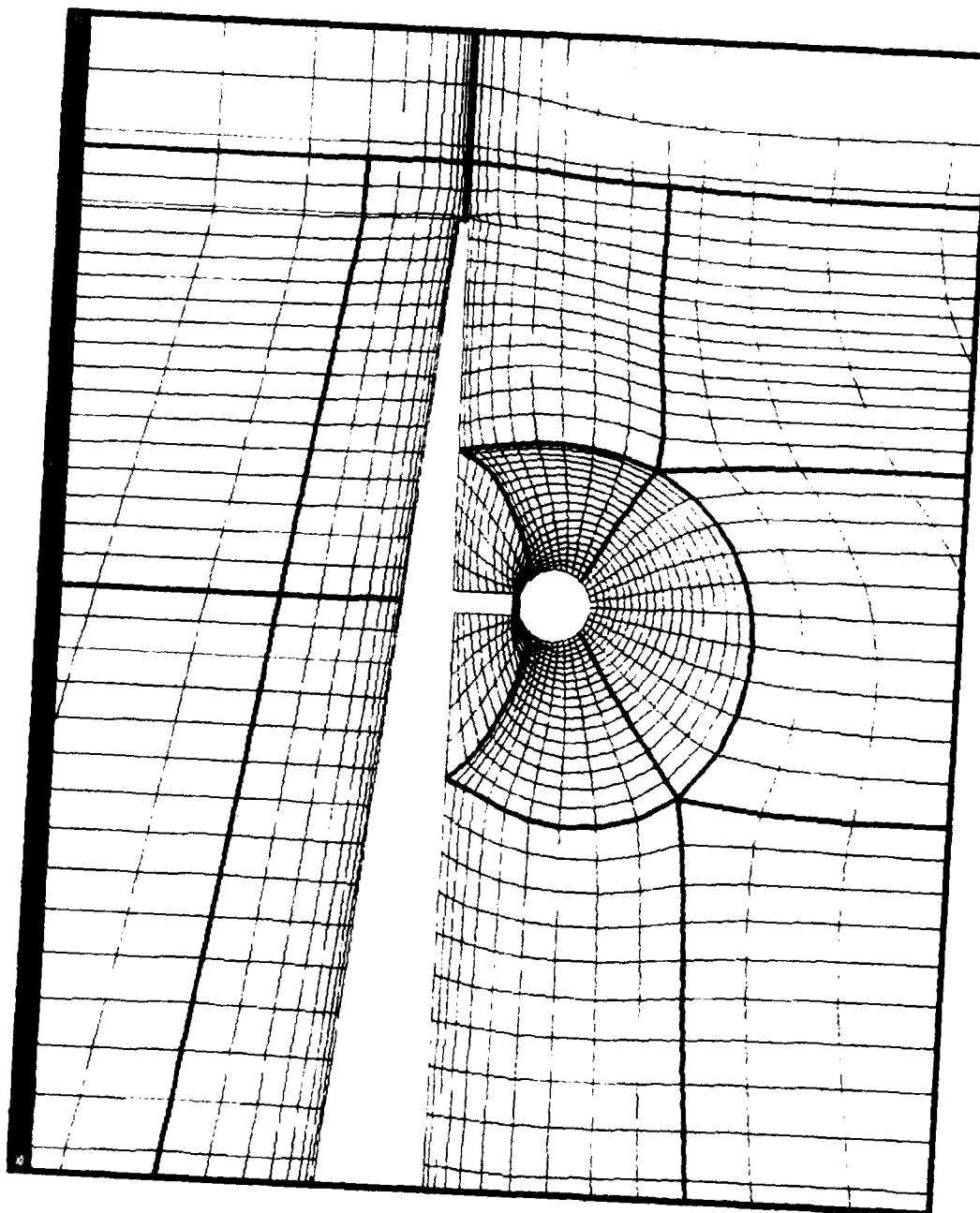
Figure 12. C-O Grid for Wing-Pylon-Store



b. Wing-Pylon-Store with Side and Back Boundaries  
Figure 12. C-O Grid for Wing-Pylon-Store (Continued)



c. Front View of Wing-Pylon-Store  
Figure 12. C-0 Grid for Wing-Pylon-Store (Continued)



d. View of Wing-Pylon-Store  
Figure 12. C-O Grid for Wing-Pylon-Store (Concluded)

The curves to be rotated each included a "C"-like segment emanating from a point ahead of the store nose and terminating at the leading edge of the wing at a point to the right or left of the pylon. A second segment of the rotated curves consisted of spar lines extracted from the lower wing grid with operation EXTRACT. The third part extended from the wing trailing edge to the downstream boundary.

Operation INSERT was then used to combine these segments, resulting in two bounding curves. Operation ROTATE was then used to form the outer boundaries of three of the five blocks in the C-O structure. Two additional blocks in the store grid structure were bounded in part by the lower wing surface. The boundaries of these two blocks ahead of the wing were generated with BLEND, using the "C" curve segments from the previous ROTATE operation as the two bounding curves. Operation TRANSUR was found to be inappropriate for generating these last two surfaces, since the bounding curves consisted of three space curves and a line degenerated to a point (a curved triangle). This logically caused the transfinite interpolation method to produce results inconsistent with the boundaries.

Figure 12c is a side view of the pylon and store, showing the interface of the C-grid with the surrounding H-grid. The difficulties inherent in interfacing the C-grid with the wing are graphically evident at the leading edge. The clustering of points in the gap between the pylon and store was necessary so that the unsteady flow field could be resolved there during store separation. A sixth block is embedded in the gap. Figure 12d is a view looking downstream at a plane about mid-chord. The asymmetry due to the swept leading edge is evident.

The remainder of the grid system, consisting of 24 blocks, surrounds the C-O system. Except for the H-O cylindrical structure ahead of the C-O system, these remaining blocks are nearly rectangular. The generation of their boundaries is remarkably easy with the EAGLE surface generating code. The far-field boundaries were placed 40 store diameters upstream and downstream of the nose of the store, and 20 diameters outboard of the store.

After all block boundaries had been generated, these surfaces were used as input to the EAGLE elliptic grid generation code. Several preliminary runs were made before the final boundary configuration was arrived at. These trial runs were necessary because of the large curvature and acute angles formed by the three-dimensional intersection of the C-O grid with the lower wing surface.

Fine adjustments of block boundary curves were necessary to prevent grid lines from crossing during interpolation for the initial algebraic grid. Even so, the grid finally deemed acceptable had crossed lines in one block in the initial grid. However, the elliptic solver corrected this in five iterations. No doubt the task would have been simplified by modifying the block structure so that the C-O grid enclosed only the store and sting, with rectangular blocks inserted between it and the wing.

The final elliptic run was for 50 iterations, requiring 526 CPU seconds on a Cray X-MP/24.

#### c. Technology Transfer

The EAGLE grid code was first released to users in the United States by the Air Force in April 1987 at a week-long workshop on its use held at Eglin AFB which was attended by 99 people. A second workshop and user's forum was held in October 1988. The code, though developed on a Cray X-MP has now been ported to a number of other systems including Cray 2, VAX, IRIS, Sun, Apollo, CYBER, and IBM. A FORTRAN 77 standard version of the code has also been produced (Reference 2). This version is interactive from a graphics workstation in the sense that input commands can be given one at a time and error messages are given immediately.

#### d. Documentation

This code was first reported in Reference 27. Full documentation for both the use of the code and its operation is given in References 18 and

19. The surface generation part of the grid code was reported in Reference 43. The adaptive version, coupled with the Euler solver, was reported in Reference 33. A number of examples of the use of the grid code have been reported in the references listed herein. Reference 45 discusses some of the specific techniques used in the application of the code.

#### 4. CONCLUSIONS

An essential element of computational fluid dynamics solutions on general regions is the construction of a grid on which to represent the flow equations in finite form. The grid must be generated for the region of interest, and this is far from being a trivial problem. In fact, at present it can take orders of magnitude more man-hours to construct the grid than it does to perform and analyze the flow solution on the grid. This is especially true now that flow codes of wide applicability are becoming available, and grid generation has been cited repeatedly as being a major pacing item. The flow codes now available typically require much less esoteric expertise of the knowledgeable user than do the grid generation codes.

The construction of structured grids in complicated regions has been greatly facilitated by the use of composite-block grids in which the region is broken up into sub-regions bounded by six (four in 2D) curved sided within each of which the grid is generated separately but with complete continuity across the connecting interfaces. This continuity is accomplished through the use of a surrounding layer of points outside each computational block, with values at the (image) points thereon set equal to those at corresponding (object) points in the interior of another (or the same) block.

This then allows both the grid generation, and numerical solutions on the grid, to be constructed to operate in a rectangular computational region, regardless of the shape or complexity of the full physical region. The full region is treated by performing the solution operation in all of the rectangular computational blocks. With the composite framework, partial differential equation solution procedures written to

operate on rectangular regions can be incorporated into a code for general configurations in a straightforward manner, since the code only needs to treat a rectangular block. The entire physical field then can be treated in a loop over all the blocks.

The original impetus for using blocked grids was to simplify the gridding of complex configurations and to permit the solution of large problems requiring many grid points by keeping only the information needed to solve one block in central memory while retaining the information associated with the remaining blocks in secondary memory. Experience exposed a third reason for using blocked grids, having to do with computer cost and/or job turnaround. Supercomputer installations generally place a high price on the use of central memory, whereas the use of secondary memory is relatively cheap. For example, on a CRAY X-MP/24, the cost increases considerably on commercial machines, and the priority decreases considerably on government machines, when more than two million words of central memory are used. By blocking, the use of central memory can be controlled, not only to fit the available memory, but also to fit the available budget.

The EAGLE grid generation system (References 1-4) is a powerful tool with which grids can be constructed for general boundary configurations. This code was designed to be very user-oriented with efficient and easily recognizable input. This algebraic/elliptic grid generation code was initially released by the Air Force at a workshop in April 1987 that was attended by 99 people from aerodynamics, hydrodynamics, electromagnetics, and other areas in industry, government laboratories, and universities. A second workshop on the code was held in October 1988 at which a second edition of the code was released. The broad range of application areas represented among the attendees at these workshops is evidence of the interest in this code and the area of numerical grid generation in general. This 35,000 line code is made available by the Air Force to users within the United States as a spin-off of the development for Air Force applications.

## SECTION IV RECOMMENDATIONS

### 1. FLOW SOLVERS

The REDOX code should continue to be exercised and maintained for the solution of transonic flow about stores. It should also be coupled with a six-degree of freedom code and used for the computation of actual store trajectories.

This research led to the development of stable algorithms capable of solving the unsteady Navier-Stokes equations on extremely fine grids that had large aspect ratios. This opportunity to investigate large aspect ratio cells led to the discovery of certain numerical difficulties when the aspect ratios were extraordinarily large. Further research into numerical difficulties associated with large aspect ratio cells is in order.

In some regards, computational fluid dynamics is ahead of wind tunnel experimentation. For example, it has, as yet, not been possible to obtain true unsteady experimental transonic store aerodynamic data. Such data are desperately needed for code validation. Even unsteady static pressure measurements on the surface of a simple store configuration as it separates, or moves relative to a flat plate would be useful. Competent experimentalists should be challenged to produce good unsteady data.

### 2. GRID GENERATION SYSTEM

Because of the emphasis on composite grids, the tasks of subdividing the grids, generating surface grids, and providing interfaces have become more time consuming and critical than the task of generating the interior grids. How a grid should be subdivided depends on the geometry, the numerical algorithm used, the flow features, etc. So, given a limited computer resource, the sub-grids of a composite grid must be selected with care. This implies a learning process and a need for human interaction. Like geometry definition, the tasks of sub-gridding,

interfacing, and surface grid definition are being assigned to interactive workstations. These are not simple tasks or ones for which off-the-shelf software is yet available. This is evidently a pacing area of research in complex grid generation.

Surface grid generation has a dominant effect on the quality of the volume grid, is very time-consuming, and is in considerable need of improvement in regard to the specification of boundary data sets and the interactive manipulation thereof. Surface definition continues to be a pacing problem. More emphasis should be put on the development of CAD geometry tools especially suited to the needs of CFD.

The topological definition of the block structure requires considerable experience and is difficult to teach. There is a need for automation of this process, perhaps through the use of artificial intelligence or other means.

The critical need for graphical interaction, especially in regard to surface grid generation, block definition, and grid control is evident. Codes should have an efficient and effective user interface with error-checking and on-line instruction. The process of grid generation for complex configurations still requires too large an amount of man-time.

It appears now that the theoretical developments necessary for effective grid generation are largely in hand, but that a very large amount of effort is still needed in the efficient implementation of the processes.

## REFERENCES

1. J.M. Janus, "The Development of a Three Dimensional Split Flux Vector Euler Solver with Dynamic Grid Applications," MS Thesis, Mississippi State University, August 1984.
2. A. Martinez, "Standardization of EAGLE Program - Numerical Grid Generation System," MS Thesis, Mississippi State University, December 1988.
3. D.M. Belk, "Three-Dimensional Euler Equations Solutions on Dynamic Blocked Grids," PhD Dissertation, Mississippi State University, August 1986.
4. B. Gatlin, "An Implicit, Upwind Method for Obtaining Symbiotic Solutions to the Thin-Layer Navier-Stokes Equations," PhD Dissertation, Mississippi State University, August 1987.
5. H.J. Kim, "Three-Dimensional Adaptive Grid Generation on a Composite Structure," PhD Dissertation, Mississippi State University, May 1987.
6. Y.S. Chae, "The Construction of Composite Grids for General Three-Dimensional Regions," PhD Dissertation, Mississippi State University, May 1987.
7. L.B. Simpson, "Unsteady Three-Dimensional Thin-Layer Navier-Stokes Solutions on Dynamic Blocked Grids," PhD Dissertation, Mississippi State University, December 1988.
8. Y. Tu, "Three-Dimensional Solution Adaptive Grid Generation on Composite Configurations," PhD Dissertation, Mississippi State University, to be published.
9. A. Arabshahi, PhD Dissertation, Mississippi State University, to be published.
10. J.F. Thompson and C.W. Mastin, "Order of Difference Expressions in Curvilinear Coordinate Systems," Journal of Fluids Engineering, Vol. 107, pp. 241-250, June 1985.
11. L.E. Lijewski, J.F. Thompson, and D.L. Whitfield, "Computational Fluid Dynamics for Weapon Carriage and Separation," AGARD Fluid Dynamics Panel Symposium on Store Airframe Aerodynamics, Athens, Greece, October 1985.
12. D.M. Belk, J.M. Janus, and D.L. Whitfield, "Three-Dimensional Unsteady Euler Equations Solutions on Dynamic Grids," AIAA 18th Fluid Dynamics and Plasmadynamics and Lasers Conference, AIAA-85-1704, July 1985.
13. J.F. Thompson, "A Survey of Composite Grid Generation for General Three-Dimensional Regions," Conference on Computational Geometry and Computer Aided Design, New Orleans, LA., June 1985; also presented as an invited paper at the International Meeting on Advances in Nuclear Engineering Computational Methods, Knoxville, TN, April 1985.

# REFERENCES (CONTINUED)

14. D.L. Whitfield and J.M. Janus, "Three-Dimensional Unsteady Euler Equations Solution Using Flux Vector Splitting," AIAA 17th Fluid Dynamics, Plasma Dynamics, and Lasers Conference, AIAA-84-1552, June 1984.
15. J.S. Mounts, A. Martinez, and J.F. Thompson, "An Analysis of Elliptic Grid Generation Techniques Using an Implicit Euler Solver," AIAA Applied Aerodynamic Conference, AIAA-86-1766-CP, June 1986.
16. J.F. Thompson, "Composite Grid Generation Techniques for General Three-Dimensional Regions," The First International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Landshut, West Germany, July 1986; and The First World Congress on Computational Mechanics, Austin, TX, September 1986.
17. J.S. Mounts, D.M. Belk, and L.E. Lijewski, "A Comparison of Explicit and Implicit Split Flux Euler Algorithms," AIAA-87-2413, 5th Applied Aerodynamics Conference, Monterey, CA, August 1987.
18. L.E. Lijewski, J. Cipolla, et.al., "Program EAGLE User's Manual, Volume I-- Introduction and Grid Applications," AFATL-TR-88-117, September 1988.
19. J.F. Thompson and B. Gatlin, "Program EAGLE User's Manual Volume II-- Surface Generation Code" and Volume III--Grid Generation Code," AFATL-TR-88-117, Air Force Armament Laboratory (AFATL), Eglin AFB, FL 32542-5434, September 1988.
20. C.J. Cottrell, A. Martinez, and G. Chapman, "A Study of Multi-Body Aerodynamic Interference at Transonic Mach Numbers," AIAA-87-0519, 25th Aerospace Sciences Meeting, Reno, NV, January 1987. (Also AIAA Journal, Volume 2b, No. 5, May 1988, pp. 553-560).
21. C.J. Cottrell and L.E. Lijewski, "A Study of Finned, Multi-Body Aerodynamic Interference at Transonic Mach Numbers," AIAA-87-2480, 5th Applied Aerodynamics Conference, Monterey, CA, August 1987.
22. C.J. Cottrell and A. Martinez, "Experimental and Numerical Data for Transonic Mutual Interference Around Unfinned Bodies," AFATL-TR-86-75, October 1986.
23. L.E. Lijewski, "Transonic Flow Solutions on a Blunt, Finned Body of Revolution Using the Euler Equations," AIAA-86-1082, 4th Fluid Mechanics, Plasma Dynamics and Lasers Conference, Atlanta, GA, May 1986.
24. L.E. Lijewski, "Transonic Flow Solutions on a Blunt, Body-Wing-Canard Configuration Using an Explicit Euler Solver," AIAA-87-2273, 5th Applied Aerodynamics Conference, Monterey, CA, August 1987.
25. A. Martinez, Y.S. Chae, and J.F. Thompson, "Applications of Numerical Grid Generation to Advanced Weapon Airframe Configurations," AIAA-87-2294, Atmospheric Flight Mechanics Meeting, Monterey, CA, August 1987.
26. A. Martinez and J.S. Mounts, "An Analysis of Elliptic Grid Generation Techniques Using an Implicit Euler Solver," AIAA-86-1766, 4th Applied Aerodynamics Conference, San Diego, CA, June 1986.

# REFERENCES (CONTINUED)

27. J.F. Thompson, "A Composite Grid Generation Code for General Three-Dimensional Regions," AIAA-87-0275, 25th Aerospace Sciences Meeting, Reno, NV, January 1987, also AIAA Journal, Volume 26, No. 3, p. 271, March 1988.
28. D.M. Belk, "Unsteady Three-Dimensional Euler Equations Solutions on Dynamic Blocked Grids," AFATL-TR-86-74, October 1986.
29. D.M. Belk and D.L. Whitfield, "Three-Dimensional Euler Solutions on Blocked Grids Using an Implicit Two-Pass Algorithm," AIAA-87-0450, 25th Aerospace Sciences Meeting, Reno, NV, January 1987.
30. D.M. Belk and D.L. Whitfield, "Time-Accurate Euler Equations Solutions on Dynamic Blocked Grids," AIAA-87-1127-CP, Proceedings of the 8th Computational Fluid Dynamics Conference, Honolulu, HI, June 1987.
31. D.M. Belk and L.B. Simpson, "Unsteady Transonic Flow Using Euler Equations," Proceedings of the Symposium on Transonic Unsteady Aerodynamics and Aeroelasticity - 1987, May 1987.
32. B. Gatlin and D.L. Whitfield, "An Implicit Upwind Finite Volume Method for Solving the Three-Dimensional Thin-Layer Navier-Stokes Equations," AIAA Paper No. 87-1149-CP, June 1987.
33. J.K. Kim and J.F. Thompson, "Three-Dimensional Adaptive Grid Generation on a Composite Block Grid," AIAA-88-0311, 26th Aerospace Sciences Meeting, Reno, NV, January 1988. Accepted for publication in AIAA Journal.
34. J.F. Thompson and L.E. Lijewski, "Composite Grid Generation for Aircraft Configurations with the EAGLE Code," in Three-Dimensional Grid Generation of Complex Configurations--Recent Progress, J.F. Thompson and J.L. Steger (eds.), AGARD-AG-309, p. 85, March 1988.
35. D.M. Belk, J.M. Janus, and D.L. Whitfield, "Three-Dimensional Unsteady Euler Equations Solution on Dynamic Grids," AIAA Journal, Volume 25, No. 9, pp. 1160-1161, September 1987.
36. C.J. Cottrell and L.E. Lijewski, "A Study of Finned, Multi-Body Aerodynamic Interference at Transonic Mach Numbers," AIAA-87-2480, 5th Applied Aerodynamics Conference, Monterey, CA, August 1987.
37. D.L. Whitfield, T.W. Swafford, R.A. Mulac, J.M. Janus, and D.M. Belk, "Three-Dimensional Unsteady Euler Solutions for Propfans and Counter-Rotating Propfans in Transonic Flow," AIAA-87-1197, 19th Fluid Dynamics, Plasma Dynamics, and Lasers Conference, Honolulu, HI, June 1987.
38. D.L. Whitfield, "Implicit Upwind Finite Volume Scheme for the Three-Dimensional Euler Equations," Engineering and Industrial Research Station Report MSSU-EIRS-ASE-85-1, Mississippi State University, Mississippi State, MS, September 1985.

#### REFERENCES (CONCLUDED)

39. D.L. Whitfield, J.M. Janus, and L.B. Simpson, "Implicit Finite Volume High Resolution Wave-Split Scheme for Solving the Unsteady Three-Dimensional Euler and Navier-Stokes Equations on Stationary or Dynamic Grids," Engineering and Industrial Research Report MSSU-EIRS-88-2, Mississippi State University, Mississippi State, MS, February 1988.
40. J.F. Thompson and D.L. Whitfield, "Transonic Flow Solutions on General 3D Regions Using Composite-Block Grids," 11th International Conference on Numerical Methods in Fluid Dynamics, Williamsburg, VA, June 27-July 1, 1988.
41. J.S. Mounts, D.M. Belk, and D.L. Whitfield, "Program EAGLE User's Manual, Volume IV--Multiblock Implicit, Steady-State Euler Code," AFATL-TR-88-117, September 1988.
42. J.F. Thompson, "A Survey of Composite Grid Generation for General Three-Dimensional Regions," in Numerical Methods for Engine - Airframe Integration, S.N.B. Murthy and G.C. Paynter, (Ed.), AIAA, 1986.
43. G.A. Jones, J.F. Thompson, and Z.U.A. Warsi, "Surface Grid Generation for Composite Block Grids," 2nd International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Miami, FL, December 1988 in Numerical Grid Generation in Computational Grid Mechanics, Pineridge Press, pp. 167-176, 1988.
44. J.F. Thompson, "A General Three-Dimensional Elliptic Grid Generation System on a Composite Block Structure," Computer Methods in Applied Mechanics and Engineering, Volume 64, p. 377, 1987.
45. J.F. Thompson and L.E. Lijewski, "Efficient Application Techniques of the EAGLE Grid Code to Complex Missile Configurations," AIAA-89-0361, 27th Aerospace Sciences Meeting, Reno, NV, January 1989.
46. R.W. Noack and D.A. Anderson, "Solution Adaptive Grid Generation Using Parabolic Partial Differential Equations," AIAA Paper No. 88-0315, January 1988.
47. B. Van Leer, Private Communication, 1983.
48. R.M. Beam and R.F. Warming, "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation-Law Form," Journal of Computational Physics, Volume 22, 1976, pp. 87-110.
49. W.K. Anderson, "Implicit Multigrid Algorithms for the Three-Dimensional Flux Split Euler Equations," PhD Dissertation, Mississippi State University, August 1986.
50. J.M. Janus, PhD Dissertation, Mississippi State University, to be published.
51. G.A. Jones, "Surface Grid Generation for Composite Block Grids," PhD Dissertation, Mississippi State University, May 1988.